UCRL-ID-114564
NRC FIN L-1867

Data Communication Systems in Nuclear Power Plants

G. G. Preckshot

May 28, 1993

FESSP
Fission Energy and Systems Safety Program

Lawrence Livermore National Laboratory

This work was supported by the United States Nuclear Regulatory Commission under a Memorandum of Understanding with the United States Department of Energy.

Data Communication Systems in Nuclear Power Plants

G.G. Preckshot
R.H. Wyman

Manuscript Date: May 28, 1993

Contents

Data Communication Systems in Nuclear Power Plants

Executive Summary

This paper surveys the state of digital communications technology as it is applied to computer communications and develops the basis for possible regulatory guidance for computer communication systems used in nuclear power plants. The focus is threefold: first, to provide an abbreviated coverage of current digital communication methods and standards; second, to review possible candidates for nuclear power plant computer communication systems and develop criteria for selection of appropriate standards and methods; and third, to develop or suggest means to verify that specific hardware and software that implement chosen standards will perform with adequate speed and reliability. The subject of multiplexing, which is often an integral part of computer communication protocols, is covered as it arises naturally in the discussion.

The building blocks of today's systems are sophisticated, competent and able to handle high-rate communications between computers and their peripheral apparatus with very high reliability. Modern communications hardware is dramatically different from that used in the nineteenth century. This reflects the improved understanding of the propagation of electrical signals in the various media, as well as vastly improved receivers and transmitters. The proper choice of medium is especially important in stressful environments such as nuclear power plants. Exposure of communications media to high levels of RF noise, high-energy electrical faults, and significant ground–mat voltage differentials are possibilities that do not exist in the usual office computer environment.

In very hazardous or noisy electrical environments, the medium of choice is fiber-optic cable because of its electrical isolation and noise immunity. In more benign environments where high data rates (greater than 2 Mbits/second) are required with good noise immunity, coaxial cable is a good choice. In electrical environments where noise and fault propagation are of less concern, twisted pair or shielded twisted pair cable is a good choice for low-to-moderate data rates (9.6 Kbits/second to 2 Mbits/second), over short to medium distances (10 m to 1 Km), and where low cost is important.

The performance of a computer network communication system is a complicated function of many elements that may be thought of as being linked in a chain. Poor performance of any link of the chain can degrade overall operation even though other elements are the best that money can buy. The most-often-made erroneous assumption is that high link bandwidth guarantees high-speed network performance. The actual measure that determines effective performance is how fast software running in one node can actually communicate with software running in another. Software can reduce performance even when all hardware concerned is fast and well matched.

Selection of protocols for use in communications systems in nuclear plants is very much dependent upon balancing safety and reliability against performance (speed and throughput) required for a specific application. For very reliable systems, which include reactor protection systems, a protocol

designed using recent results in protocol engineering may be the only way to obtain sufficient assurance of safety. Use of a standard protocol is not a guarantee of suitability. Standard and proprietary protocols exist whose specifications have not been validated by formal techniques and which may contain specification errors.

A variety of standard and proprietary protocols are reviewed in this paper and their more important characteristics are discussed. Fifteen useful questions are posed for reviewers of communications systems proposed for nuclear power plant use. Thirteen design elements that complicate safety and reliability determinations are presented, and fourteen complementary design elements that improve the ease of safety determination are presented with an explanation of their effects.

In one respect, communications protocols are no different than any other software. The same need exists for ordered and controlled software development methods as for any highly reliable software. Because of the very subtle interactions of communicating entities and because protocols are susceptible to formal descriptions, a body of work has grown that is particular to communications protocols. Formal methods were employed in 1980 or earlier, and two international standards bodies (ISO and CCITT) have cooperated to standardize several Formal Description Techniques (FDTs). The protocol specification and validation process is now fairly well understood, and recent recommendations propose use of formal methods. The essence of design is to synthesize a protocol having desired functions, describe it unambiguously, and prove that the hazards mentioned in later sections of this paper do not exist.

Organized development of computer network protocols began in the late 1960s, and since the late 1970s there has been a steady increase in techniques and tools available for protocol synthesis, validation, and verification and for network design. A sampling of significant tools and techniques is described in the body of this paper.

We conclude that currently available media, equipment, and protocol development and validation techniques are adequate to the task of developing a multiplexed communication system suitable for use in reactor protection systems if the tools and techniques are professionally and carefully applied.

1. Introduction

The purpose of this paper is to survey the state of digital communications technology as it is applied to computer communications and to develop the basis for possible regulatory guidance for computer communication systems used in nuclear power plants. The focus is threefold: first, to provide an abbreviated coverage of current digital communication methods and standards; second, to review possible candidates for nuclear power plant computer communication systems and develop criteria for selection of appropriate standards and methods; and third, to develop or suggest means to verify that specific hardware and software that implement chosen standards will perform with adequate speed and reliability. The subject of multiplexing, which is often an integral part of computer communication protocols, will be covered as it arises naturally in the discussion.

The next sections introduce the building blocks that are used in any digital communications system, together with existing standards and their applications. A discussion of the use of these blocks and standards in nuclear power plant systems follows, and key questions are presented. Issues that need to be considered in any design are presented and, finally, issues related to regulatory guidance are considered.

A glossary is provided to explain unusual or computer-specific terminology.

2. Fundamentals

Digital communications systems have evolved from seeds planted by Morse, Edison, and others. From the simple telegrapher's key in the nineteenth century through the Teletype in the early twentieth century, that evolution has continued, resulting in today's elaborate systems and protocols. The building blocks of today's systems are sophisticated and competent and can handle high-rate communications between computers and their peripheral apparatus with very high reliability. The following section discusses those building blocks.

2.1 Hardware

Communications hardware of today is dramatically different from that used in the 1950s. This reflects the improved understanding of the propagation of electrical signals in the various media and vastly improved receivers and transmitters. But there are still vestiges of early twentieth-century practice in today's applications, even though they are sometimes hard to discern. For instance, start and stop bits found in most modern asynchronous serial communications originally were used to start and stop the motor-driven shaft of a rotary selector in teleprinter machines.

Besides the terminus hardware (the sending and receiving electronics), the communication system designer has available the choice of physical width of the communication channel (serial or parallel) and the medium over which the signals will be transmitted.

2.1.1 Serial Transfer

The serial bit stream is the main method of data communication today between computers, computers and terminals, and between computers and other entities because of distance and expense.

Where data transfer can go on in both directions simultaneously, the term “full duplex” is used to describe the communications “channel.” Where data transfer can go on in both directions but not simultaneously, the terms “half duplex” or “semi-duplex” are used.

Most national and international standards concerned with computer data communication employ serial transfer.

2.1.2 Parallel Transfer

For some applications, parallel transfer is employed. In this type of transfer, multiple bits are transferred from transmitter to receiver at each clock time. Typically this type of transfer is between a computer and a high-speed data store such as a disk, but in some applications it is used to transfer data between computers. These transfers occur over relatively short distances but at very high data rates (about 1 gigabit per second) and are usually used between closely connected, very-high-speed computers. Parallel transfer between computers is unlikely to be used in computer networks employed in process control systems or in nuclear reactor protection systems because distances are longer than usually encountered in parallel transfer applications, electromagnetic interference (EMI) and RFI noise sources are more prevalent, and the data rates required are lower. For this reason, with the exception of instrument buses such as IEEE-488, parallel data transfer will not be considered further in this report. Any proposal to use parallel communication in safety systems should probably receive heightened scrutiny.

2.1.3 Media

The proper choice of medium is especially important in stressful environments such as nuclear power plants. Exposure of communications media to high levels of RF noise, high-energy electrical faults, and significant ground–mat voltage differentials are possibilities that do not exist in the usual office computer environment. Media also vary considerably in the data rate that they can handle.

Where conductors are used for high-rate signaling, twisted pair cable or coaxial cable is the medium of choice. Pulses propagate through these media at about 1.5 nanoseconds per foot, whereas in free space, signals propagate at about one nanosecond per foot.

Good fidelity can be achieved in twisted pair cable if distances are short and the signal frequencies are not too high. Distance and frequency trade off, so that low-frequency signals can be faithfully reproduced over long distances while high-frequency signals may only be transmitted over short distances. Typical impedance of unshielded twisted pair cable is approximately 100 ohms.

Coaxial cable provides much better fidelity but at greater expense than twisted pair cable, provided the quality of the cable is good. Electrical losses must be kept low (low resistance in the conductor and low leakage in the dielectric). An ideal loss-less cable would provide perfect fidelity at the receiving end after the delay of the cable. Typical cable impedances range from 50 to 100 ohms.

Noise can be introduced in conductor-type media by discontinuities in conductors or dielectric or by EMI. Connectors and terminations introduce such discontinuities, particularly “tapping,” which is a common method of making connections by drilling into the cable. At a tap, signal reflections will occur that appear as noise. Noise can also be introduced by failing to terminate cables with the characteristic impedance or with a clipper. In spite of the noise potentially introduced by taps, this connection method is favored for Ethernet networks because of its convenience.

In comparison with conducting media, fiber-optic cable has many desirable characteristics and some disadvantages. Much higher rates are easily achievable using fiber optics rather than coaxial cable. Further, fiber-optic cable provides electrical isolation and immunity to external noise, which is difficult to achieve using conducting media. However, fiber-optic cable cannot be easily tapped and considerably more precision is required to attach connectors. In spite of these drawbacks, it is becoming the medium of choice in industrial applications where isolation, noise immunity, and high data rates are important.

2.2 Bits, Characters, Frames, and Blocks

Although digital information can be transmitted in the form of multiple electrical or optical levels over the media mentioned above, in practice the basic form of transmission is *two* levels or transitions between levels, intensities, frequencies, or phases. The method of varying the one or more electrical or optical energy parameters to convey information is called "modulation." The two-valued increment of information conveyed by the modulation in a serial stream is called the "bit."

Single bits by themselves are not terribly informative. Thus, bits are usually combined in message units called "characters,"[1] and larger accumulations of bits or characters are called "blocks" or "frames." Characters are typical of earlier communications methods, which essentially transmitted the written word character by character. Inefficient transmission and difficulties with timing and synchronizing high-speed communication links with character data led to the use of frames. Frames spread the overhead of timing and synchronization over larger collections of bits and permit more effective error detection and correction. Most recently developed serial communication systems use some sort of frame transmission rather than raw character data, although a frame can contain characters within it. The term "packet" is sometimes used synonymously with frame.

2.3 Synchronization

In all serial communications systems, the receiving end must be in step with the transmitting end in order to decode the incoming bits from the modulated signal. This process is called "synchronization," and the components that are synchronized are called the "transmitter" and "receiver" clocks. Clocks are electronic oscillators that provide the basic bit timing for the receiver and transmitter circuits.

2.3.1 Asynchronous Clocking

In asynchronously clocked systems, the period over which the clocks must stay synchronized is one character, typically ten bits or less. At the beginning of each character there is a "start" bit that signals the receiving end that a character is starting, at which time the receiving end re-synchronizes its clock. The receiving clock is then stable enough to remain essentially "in sync" through the remaining bits of the incoming character, which is terminated with a "stop bit." The character is a fixed length so the receiver knows when the end is reached. This system was developed for telegraph and teleprinter operation and was the staple of early computer communication systems. The UART (Universal Asynchronous Receiver Transmitter) integrated circuit was developed for this type of interface, and is still used extensively in computer systems for connecting terminals and for low-cost computer-computer connections.

2.3.2 Synchronous Character Clocking

Asynchronous signaling is relatively inefficient since there are "start" bits and "stop" bits that carry only synchronizing information. These bits are eliminated in synchronous signaling systems.

[1] A character is usually 7, 8, or 9 bits. The computer industry seems to have settled on the byte or octet (8 bits) as the size standard by which it measures data volume.

In synchronous-character-oriented systems (Conard 1980), the receiver extracts the clock from the incoming signal.[2] If no signal is present, the receiver clock operates at a default rate that is close to the expected rate. When the transmitter starts sending, initially it sends "sync" characters that are used by the receiver to synchronize and lock its internal clock. The first non-sync character is considered to be the start of the message proper. If the receiver clock loses synchronization lock due to noise or instability, the receiver must again wait for the transmitter to send sync characters.

Synchronous-character-oriented systems are rarely used for new designs because of the advantages of the method to be described next.

2.3.3 Synchronous Frame Clocking

The sync character and a number of other link control characters have bit patterns that can also represent valid binary data. Because of this and because of improved synchronization performance, "bit-stuffing" synchronous serial communication was developed (Carlson 1980). Frames are transmitted separated by "frame markers," which are special bit sequences that can never represent valid data. By hardware insertion and removal of zero bits (bit-stuffing), no valid data is transmitted that could imitate a frame marker. Bit-stuffing will be more extensively described in Section 3.2.

A recent alternative coding method is called N of M bit coding ($N < M$)[3] and is used in the FDDI (Fiber Distributed Data Interface) standard (Ross 1986). Some form of bit-stuffing or N of M bit symbol coding is used for most modern serial communication link designs.

2.4 Timing

Synchronization refers to timing on a bit-by-bit, character-by-character, or a frame-by-frame basis. There are larger issues of timing in practical communications systems that relate to the timing of messages and transactions taking place over the system. A basic issue is whether information will be transmitted as it becomes available or transmitted periodically. Another concern is whether a message offered for transmission will be delivered within a guaranteed maximum time. These attributes have significant bearing on the reliability and the suitability of a communication system for its intended application.

2.4.1 Event vs. State Timing

Two models often used for designing control systems have their counterparts in data communications. These are the event model and the state model, which can be loosely described as asynchronous system design versus synchronous system design. A system designed to the event model will send messages whenever it has something to communicate. A state model system communicates "state" at regular, sometimes precisely defined, intervals. "State" is the entire set of data shared between the communicants whether the data has changed or not. Both the event model and the state model should be regarded as ideals that are approached in practice. Actual systems may display some attributes of both models, with a preponderance of one or the other.

State model designs have the important characteristic that the communication load is constant no matter what the system is doing, thus eliminating the possibility of overloads. The cost for this feature

---

2 Usually with an electronic circuit called a phase-locked loop.

3 Code units consist of M-bit symbols, of which the equivalent of N bits represent valid data symbols, while the remaining symbols are link control symbols. This means that there are 2^N data symbols and $(2^M - 2^N)$ link control symbols. These are still frame-oriented systems, and one of the link control symbols is the frame marker symbol.

is that state model designs make poor use of communication bandwidth, since most of the time much of the data being sent is unchanging.

An event-based system transmits information when changes occur. Event model designs use the communication system bandwidth more efficiently, but have the potential for congestion and overloads during peak load periods. This type of system is relatively quiet when in steady state, but the amount of data transmitted can overwhelm the system in the event of a severe upset.

2.4.2 Determinism, Throughput, and Delay

The following discussion applies to multiple-access communication systems. These systems have more than two communicants contending for bandwidth on a shared medium. Examples include Ethernet, the IEEE 802 networks, various proprietary "multi-drop" networks, and FDDI.

A "deterministic" communication system delivers messages within a finite, predictable time delay that is a function of system communication load. Determinism is considered to be a very important quality for communication systems used in real-time applications. Unfortunately, no communication system is deterministic under all conditions, and unpredictable delays will be incurred for at least some errors or failures.

A non-deterministic communication system delivers messages with a probability of success that increases asymptotically toward one as message time delay increases. This means that there is a non-zero probability of failure-to-deliver even at very long message delivery delays. Some messages are never delivered. In spite of this, a lightly-loaded non-deterministic communication system may perform as reliably as a deterministic one because the probability of failure-to-deliver at light loads approaches the probability of system failure, which neither deterministic or non-deterministic systems can tolerate.

There are relations between "throughput" (the amount of data actually transmitted), "offered load" (data offered to the system for transmission), and message delay. Non-deterministic systems exhibit initially increasing throughput and delay with increasing offered load. "Unstable" systems collapse at a critical load (Kleinrock and Tobagi 1975, Tobagi 1980) with throughput going to zero and delay going to infinity. Figures 1 and 2 illustrate this collapse. Stable non-deterministic systems such as Ethernet (Shoch and Hupp 1980) do not show this (Figure 3). Deterministic systems show increasing throughput and moderately increasing (but limited) delay that is a predictable function of offered load (Ciminiera et al 1988, Strayer and Weaver 1988) (see Figure 4).

2.5 Protocol Software

The IEEE defines a protocol as "a set of conventions that govern the interaction of processes, devices, and other components within a system" (IEEE 1990). A communications protocol is the detailed set of rules, specifications, and timing relations that is used to implement a communications model. A model does not require specific voltages, timings, message formats, or the like. A protocol implementing a model does, so that two or more processes adhering to the protocol will successfully enact the model when interacting with each other. A number of papers (Tanenbaum 1981, Pouzin and Zimmermann 1978, Sproull and Cohen 1978) and numerous books (Davies et al 1979, McNamara 1977) have tutorials on protocols. Although protocols are often thought of in terms of software, they deal with hardware interactions, too.

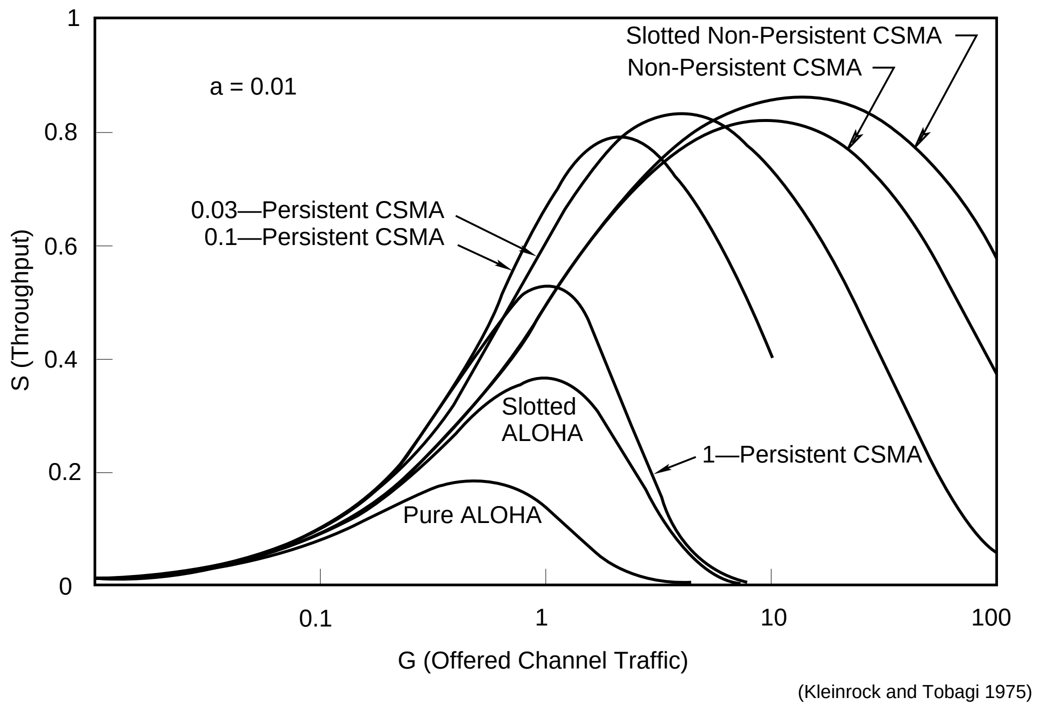


Figure 1. Unstable systems collapse under a critical load.

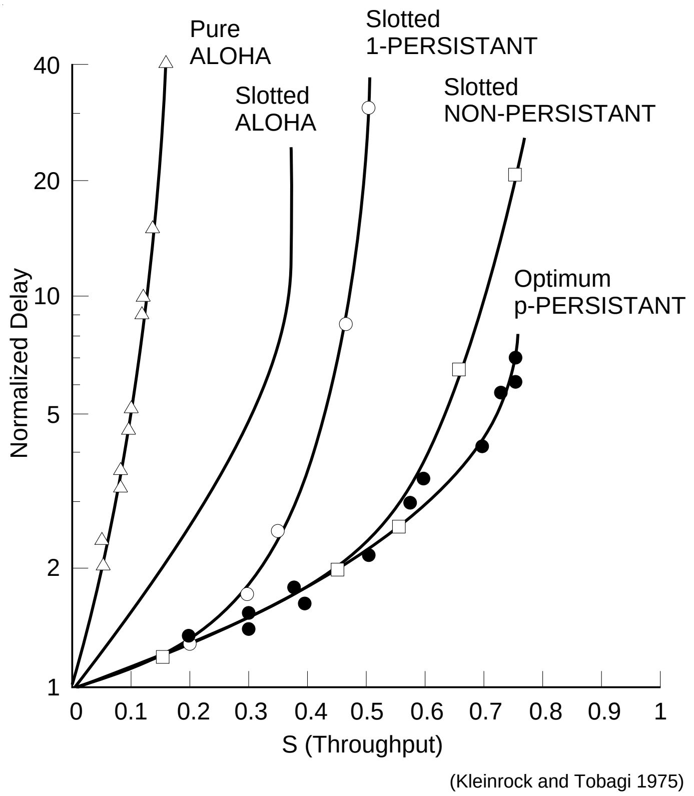


Figure 2. Throughput drops to zero and delay increases to infinity during collapse of the system.

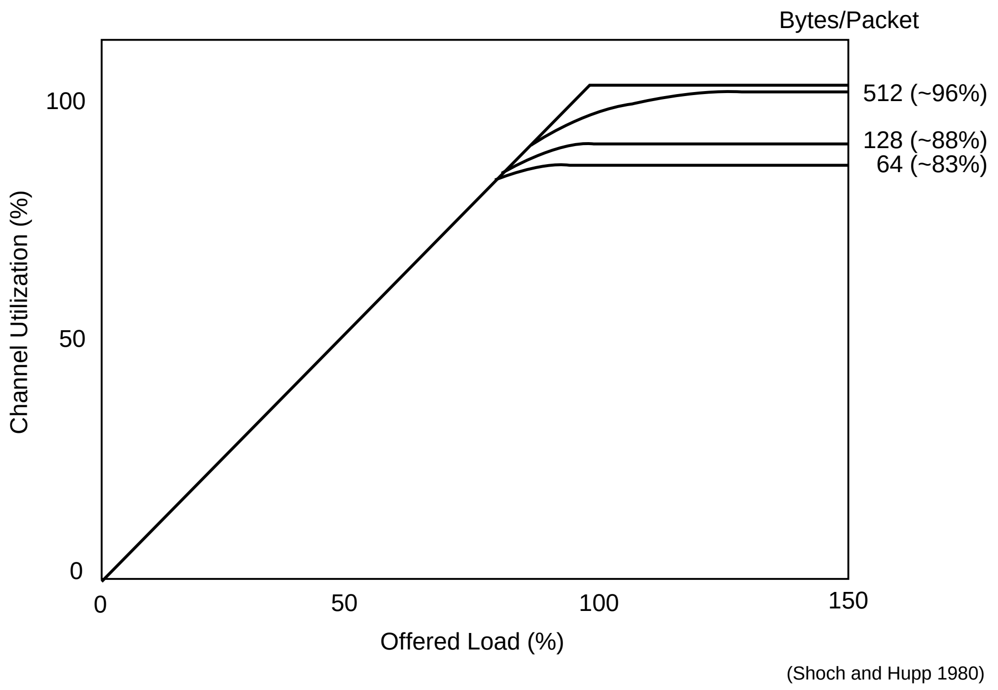


Figure 3. Stable, non-deterministic systems do not collapse under a critical load.

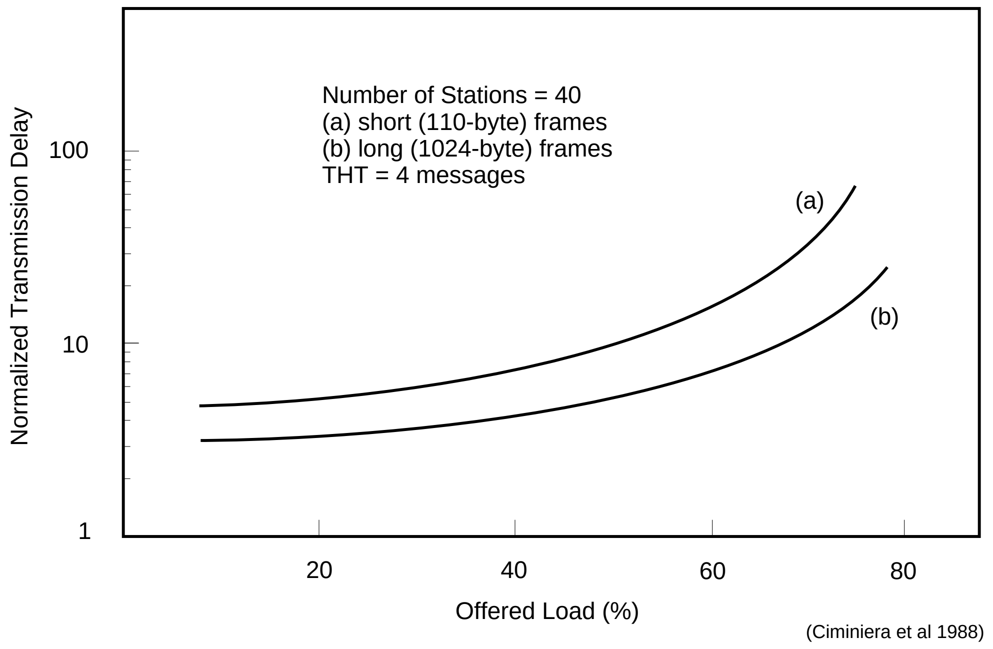
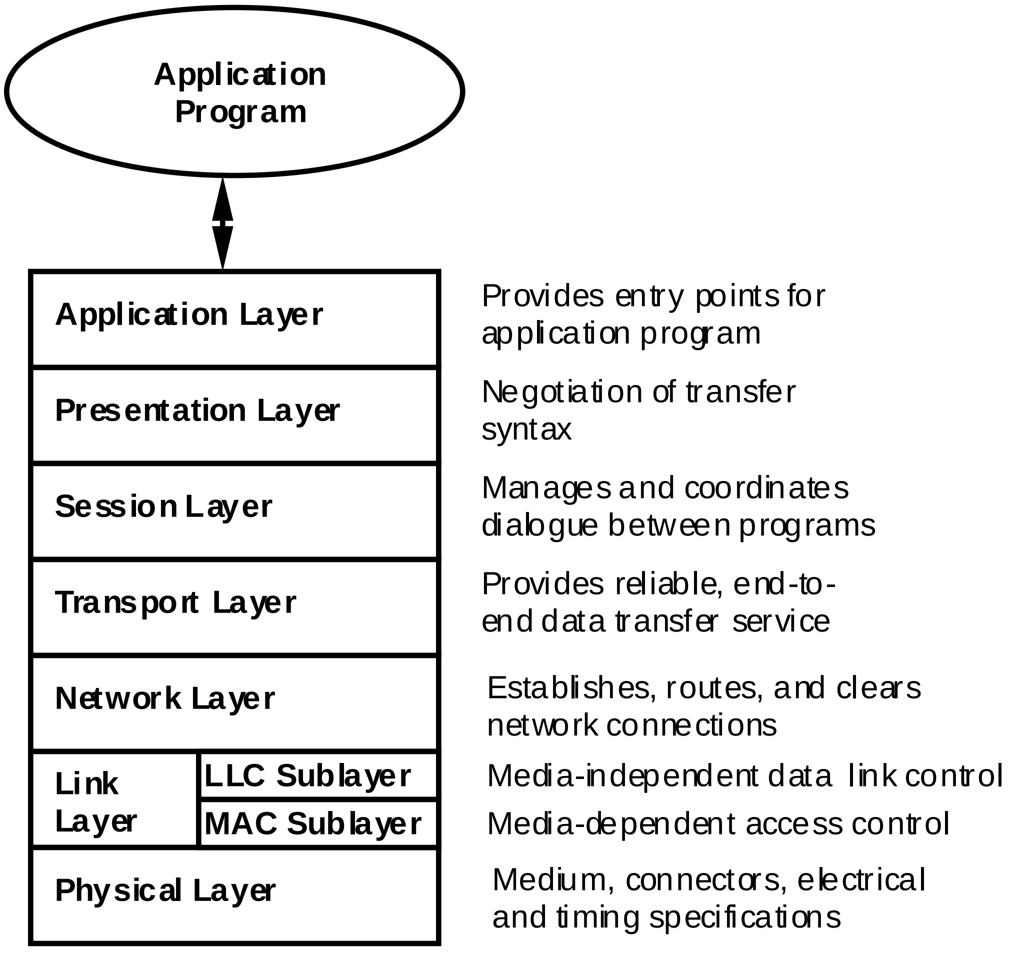


Figure 4. Deterministic systems show increasing throughput and moderately increasing delay.

2.5.1 ISO Seven-Layer Model

In 1979 (ISO 1979) the International Standards Organization (ISO) published a seven-layer Open Systems Interconnect (OSI) model for organizing emerging network communication protocols (Zimmermann 1980, Modiri 1991). This model separates various functions of communication network protocols into layers so that there is an agreed concept of what functions are to be performed and what the interfaces are between these functions. Besides layering, the OSI model introduces a second important idea that will be discussed in the sequel: encapsulation. These concepts are outlined below. Figure 5 shows the various layers of the model and the functions performed at each level. The applicability of this model and specific implementations ("protocol stacks") are also discussed.



Application Program
Application Layer
Provides entry points for application program
Presentation Layer
Negotiation of transfer syntax
Session Layer
Manages and coordinates dialogue between programs
Transport Layer
Provides reliable, end-to-end data transfer service
Network Layer
Establishes, routes, and clears network connections
Link Layer
LLC Sublayer
Media-independent data link control
MAC Sublayer
Media-dependent access control
Physical Layer
Medium, connectors, electrical and timing specifications

Figure 5. Layers of the OSI model and functions performed at each level.

2.5.1.1 Layering

Layering has two advantages: separating functions and providing well-defined interfaces between layers. The intended effect of layering is to make protocols at one level of the OSI model independent of protocols at other levels. The model has been reasonably successful in accomplishing this goal.

Some of the layers have been divided into sublayers as the understanding of physical network communications problems increased. In particular, the IEEE 802 project subdivided the ISO link layer into the Medium Access Control (MAC) sublayer and the Logical Link Control (LLC) sublayer, as shown in Figure 5. The MAC sublayer actually includes the physical layer because it was discovered that some link control issues were inseparable from the hardware being used.

The lower four layers of model protocols are devoted to providing reliable message transportation and operate independently of the end user. The upper three layers are devoted to user needs and use the reliable transportation services supplied by the lower layers.

2.5.1.2 Encapsulation

Encapsulation is the idea that protocols at succeeding lower levels of the OSI model wrap or encapsulate data coming down from higher levels with information that the lower protocols need to perform their functions. This is illustrated in Figure 6. Typically, each protocol layer adds a header containing information that will be used (and removed) by its counterpart at the receiving end. To maintain independence between OSI layers, no protocol is supposed to make assumptions about or use data in any part of the message except the encapsulation that it does itself.

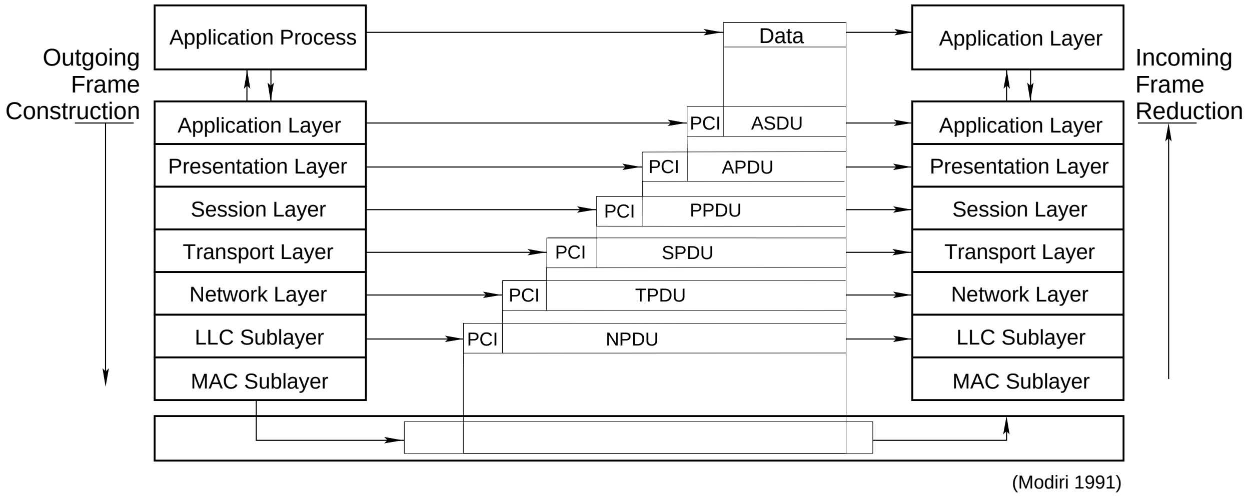


Figure 6. Protocols at succeeding lower levels of the OSI model encapsulate data coming down from higher levels with information that the lower protocols need to perform their functions.

2.5.1.3 Protocol Stacks

The software that makes up an implementation conforming to the OSI model is called a "protocol stack," taken from the usual pictures, such as Figure 7, that are used to illustrate the model. This paper will use the term in following sections.

A particular group of protocols that make up a protocol stack for a well-known application area such as Manufacturing Automation Protocol (MAP) is known as an "ISO roadmap" (Modiri 1991).

2.5.1.4 Applicability

The full OSI protocol stack may not be appropriate for some purposes. A notable example is a local area network used to connect real-time manufacturing equipment. For performance reasons, it may be necessary to omit the network layer through the presentation layer, since the functions of these layers are unnecessary and reduce performance if they are used in this situation. MiniMAP (to be discussed in Section 3.6) is an example of such a truncated protocol stack.

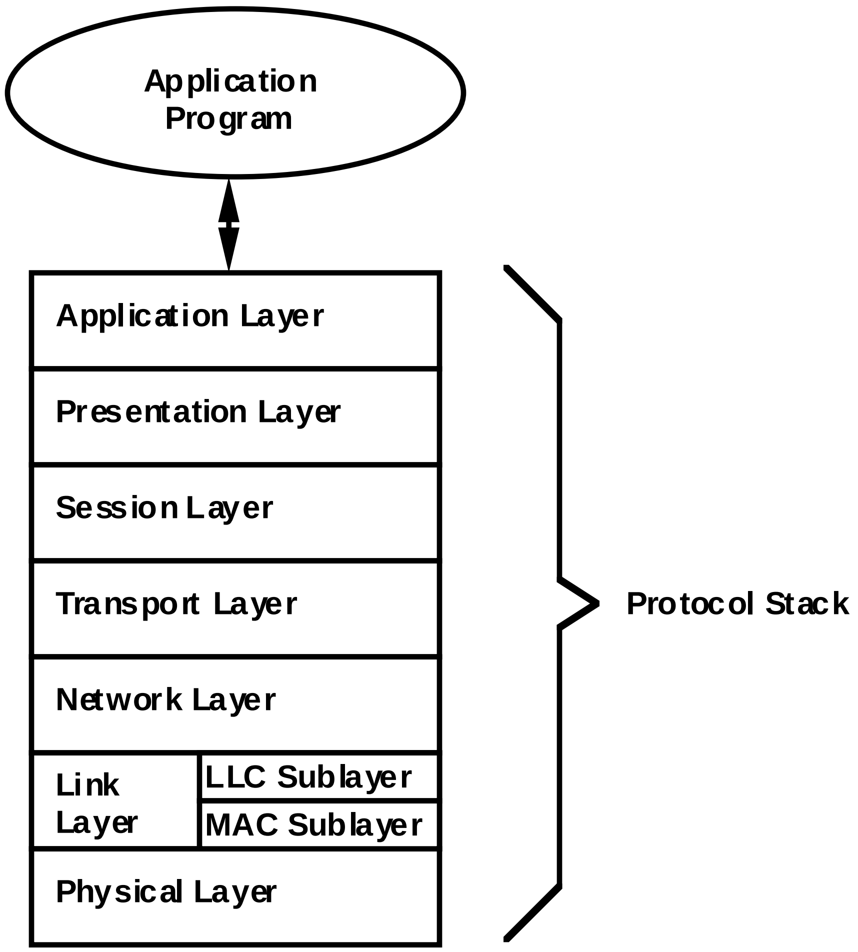


Figure 7. Protocol stack.

2.5.2 Multiplexing

From the time of Baudot in the 1870s, multiplexing communications (sending more than one data stream simultaneously over a single physical link) has been an essential part of the industry. A number of standard multiplexing methods that were developed over the years have been applied to telecommunications and radio, but none fitted so naturally to their media as the concept of multiplexing fits with computer network protocols. In short, the primary job of many multi-access computer network protocols is to multiplex many computer data streams on a few physical transmission media.

Early multiplexing methods evolved out of the need to transmit many channels of character data simultaneously or many voice channels simultaneously. These methods are still used in situations where they are appropriate, and some instances are noted in the TDM and FDM descriptions that follow. Packet multiplexing, the new method brought in by computer networks, is also described.

2.5.2.1 Time Division Multiplexing (TDM)

Time division multiplexing was an early form of multiplexing because mechanical rotary commutators[4] could be built and were easy to understand. Later incarnations use electronic means of switching. The method is still used extensively and is even proposed for new standards, such as FDDI II (Ross 1986, Ross 1989). In its simplest form, TDM consists of transmitting frames at precisely regular intervals, with sub-intervals within the frame dedicated to individual data channels. If, for instance, a frame is divided into n-bit sub-intervals, then each data channel is assigned a bandwidth of n bits every frame time. FDDI II uses essentially this approach to multiplex as many as sixteen 6.144 Mbit/second data channels on a 100-MHz physical data link.

TDM is very attractive for state model communication systems, and is often chosen for deterministic real-time systems for this reason.

2.5.2.2 Frequency Multiplexing (FDM)

Another form of multiplexing is mapping the spectra of several low-frequency signals into adjoining spectral bands in a higher frequency “broadband” signal. The single broadband signal is then transmitted over a high-frequency communication channel and the individual signals are de-multiplexed at the other end by reversing the spectral mapping.

FDM is often used in microwave communication systems and high-capacity broadband communication systems. An example of this is multiplexing several local area networks (frequency range 10 MHz) on broadband cable at many times the network frequencies.

Frequency Shift Keying (FSK) is a variant of FDM which maps two binary data streams to audio frequencies suitable for transmission over ordinary telephone lines. The lowly modem uses FSK to connect computers over commercial telephone circuits. FDM is not generally used in real-time control systems except for low-speed remote units connected by inexpensive telephone equipment.

2.5.2.3 Packet Multiplexing

The term “packet” (a frame) was apparently applied to communication by Davies (Roberts 1978) but the concept of “store and forward packet communication” was first proposed by Baran (Baran 1964). A seminal influence of packet communication was that it decoupled the idea of multiplexing from the fixed bandwidth allocation inherent in the TDM and FDM approaches of that time. Rather than dedicating fixed time slots or frequency bands as in the previous two methods, packets carry data stream identification within themselves, allowing continuously variable, on-demand allocation of bandwidth to individual data streams as needed.

[4] A rotary mechanical switch.

Packet multiplexing fits the event model of communication most closely, and it is the method of choice in most office computer communication networks today. A rich set of protocols is available to service the many varied needs of office and nationwide data networks. Some of these protocols can be used in real-time systems, but they are usually considered non-deterministic and are not used in critical hard real-time systems.

2.5.3. Error Detection

No communications medium is completely error free, so error detection is the key to reliable communications. If an error occurs and is detected, steps can be taken to correct it. Error-detection schemes are based upon the transmission of redundant information and the detection of inconsistencies in the received data. The qualities of an error detection scheme are based on trading off two antagonistic factors: minimizing the amount of redundant information transmitted versus maximizing the error-detection coverage. Efficient designs strive to minimize redundant information while detecting as many errors as possible.

One particular error-detection method is notable because of its widespread use in many communication protocols. This is the Cyclic Redundancy Check (CRC) method. A 16- or 32-bit check sequence is calculated by modulo-2 division of the message or frame by a binary polynomial (McNamara 1977, Chapter 13). The check sequence is appended to the outgoing message. At the receiving end, the calculation is performed again and a mismatch with the terminating check sequence signifies an error. The various protocols that use CRC differ only by the polynomial chosen for the calculation.

CRCs are used for serial communications because of their sensitivity to "bursts" of error bits, a common type of noise encountered in serial or multi-access connections. Hardware (chips) exists that computes CRCs as the data arrives, making the method applicable to time-critical real-time systems.

2.5.4 Methods of Error Correction

Once an error is detected, the next step is to handle it so that ultimately no error is sent to the application. There are a number of effective techniques, all of which involve more redundancy in the channel. It should be kept in mind that if the error rate is high enough, the channel may become unusable simply because there is so much redundant traffic.

2.5.4.1 Disregard

In some state model systems, the appropriate course of action may be to simply disregard the erroneous message or frame and wait for the next.

2.5.4.2 Repeatback

Repeatback simply involves the re-transmission of any message or frame received by a node back to the transmitting node for verification. If the message is received at the transmitter exactly as sent, then the transmitting station has direct knowledge of successful receipt.

2.5.4.3 ACKing and NAKing

The repeatback technique doubles the traffic on the network and half of the traffic is redundant. If the receiving station checks the message for errors using embedded check bits, it can reply to both successful and unsuccessful transmissions with a much shorter message. A message can be acknowledged (ACK) if successfully received or not acknowledged (NAK) if an error has occurred. If an ACK is received at the transmitting node, the transmitter then knows it can send the next message. This is a form of flow control. If the transmitter receives a NAK, it then knows to re-transmit the message. If either the ACK or NAK gets lost, then the transmitter will have to re-transmit after a timeout. Naturally, the receiver has to be prepared for duplicate messages and the protocol used has to allow for some way for the receiver to recognize a duplicate without having to make elaborate comparisons or keep extensive records on messages received.

The use of NAK messages has a significant problem: if the receiver cannot decode a message due to errors, it cannot know to whom or when to NAK. Largely for this reason the error control method described below is used more often.

2.5.4.4 *Re-Transmission After Timeout*

In this method, the receiving node checks the message using the redundancy bits, and if it finds an error, it throws the message out. If the transmitting node has not received an ACK after a time period (timeout), it re-transmits the message. Again, the receiver must be prepared for receiving duplicates in the event the ACK gets lost.

2.5.4.5 *Forward Error Correcting Codes*

In the late 1940s Richard Hamming of Bell Laboratories invented the first error correcting code, the Hamming Code. In this work he showed how to include sufficient redundancy in a string of bits that any single error in the string, including the redundant bits themselves, could be corrected at the receiving end without re-transmission. He went on to prove the existence of multiple-error correcting codes, but he was unable to create one. He did show, however, that a single-error correcting code could be modified so that, in addition to correcting single errors, it could also detect all double errors.

In later years, the work he started was extended to multiple error correcting codes and burst-error correcting codes. All of this is available to the communications protocol designer today. However, these codes require considerable software or hardware for recovery of the data in a message with errors, and the economics of modern channels dictate that in most cases the technique of detection with re-transmission is sufficient to accomplish the task.

2.5.5 Undetectable Error Rate

In any communications system containing random noise, there is a non-zero probability that a message will get through all of the error detection and correction equipment and still contain an error. If no error-handling equipment is in place, the undetectable error rate (UER) is simply the bit error rate. But the calculation of UER from theoretical considerations for systems containing error-detection equipment is extremely complex and in most cases does not give good results in practice because the statistics of the channel are not well known. In some cases, an upper bound can be calculated on UER, but even this is very difficult.

2.6 Data Communications Performance

The performance of a computer network communication system is a complicated function of many elements that may be thought of as being linked in a chain. Poor performance of any link of the chain can degrade overall operation even though other elements are the best that money can buy. The most-often-made erroneous assumption is that high link bandwidth guarantees high-speed network performance. The actual measure that determines effective performance is how fast software running in one node can actually communicate with software running in another. Transmission of data across the physical link(s) is only one of several things that have to happen before two programs have communicated. If performance standards include reliability goals and error recovery requirements, a very large number of things may have to happen before two programs have communicated.

2.6.1 Factors That Limit Performance

Link bandwidth is clearly a limitation on data transfer rates between two nodes of a network. However, other hardware and software may place lower limits on data rates (Figure 8). Network interfaces may be unable to deal with one message immediately following another, and bus bandwidths within the node processor may limit data transferred to node memory to a fraction of that of which the network medium is capable. Once received, messages still must be processed. A slow CPU will not handle many messages per unit time regardless of how fast messages are placed in memory.

Software can reduce performance even when all hardware concerned is fast and well-matched. The protocol stack may be inefficient or may spend too much time because it provides services that are unnecessary for the application. The operating system used may spend too much time in overhead functions at the expense of communication speed and application-task CPU time. Resource contention and task synchronization overhead may delay the execution of application-task software receiving the message, so that even though the message is available, the software that needs it cannot run. Finally, the application software that uses the communication services provided by the operating system, the protocol stack, and the hardware may use those services inefficiently, resulting in far worse performance than might have been predicted.

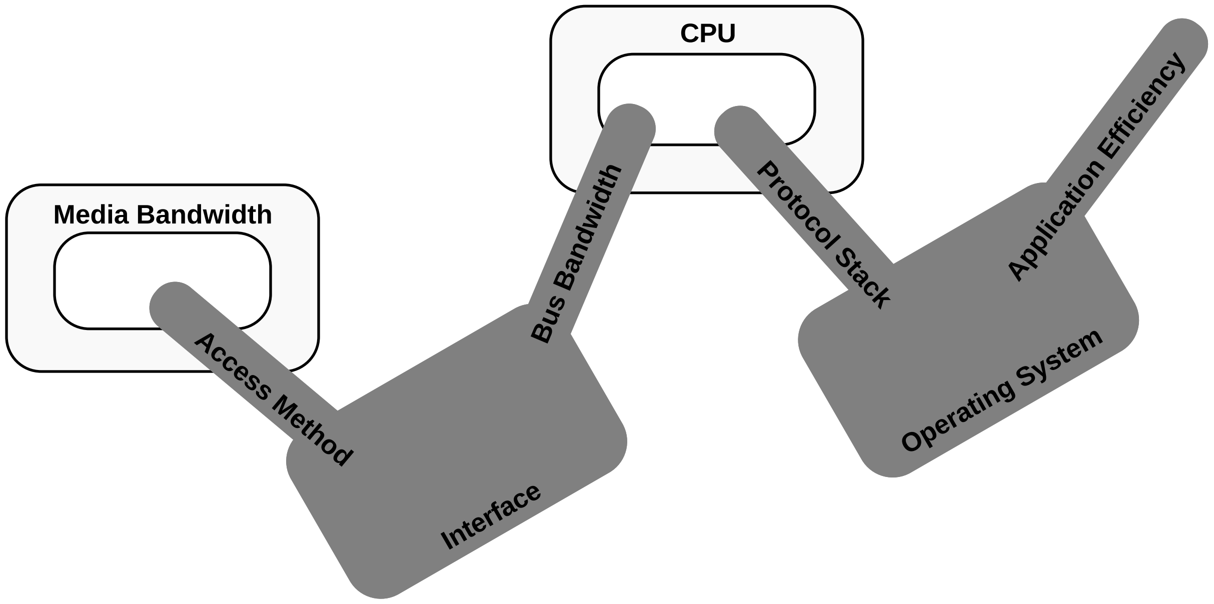


Figure 8. As well as link bandwidth, other hardware and software may limit data rates.

3. Existing Communication Methods

3.1 Traditional Serial

Serial communications links transmit a serial stream of bits in one or both directions, with the traditional approach supporting one conversation or channel.[5] A physical link may be half- or full-duplex, supporting in turn a half- or full-duplex channel. Traditional serial links have their roots in telegraph and telephone practice, specifically in teleprinter practice. A significant problem with traditional serial links is their lack of "data transparency," which means that link control characters can be mistaken for valid binary data. The reverse (binary data mistaken for link control characters) is also true.

3.1.1 Asynchronous Character Serial

Asynchronous serial communication links can start and stop on every character, which means that at least two extra bits accompany each character.

3.1.1.1 RS-232/CCITT V.28

In 1969 the Electronic Industries Association (EIA) promulgated a voltage, timing, and line definition standard for connecting data terminal equipment (DTE, the terminal) and data communication equipment (DCE, the modem). This standard, the widely recognized RS-232 serial line standard (Bertine 1980), is often taken to include ASCII character set encoding with several conventions regarding character meaning. In fact, the standard addresses only voltage levels, data rate, timing, and line definition.

In spite of its age, the RS-232 standard is used extensively today for connecting video display terminals (VDTs) to modems and computers, and computers to modems and other computers. The design of the UART integrated circuit in the 1970s made the design and production of serial computer interface circuits inexpensive, with the result that RS-232 interfaces may be the least expensive and most pervasive computer communications interface available today. In consequence, RS-232 communication links are usually found in communications systems that are cost-sensitive with low performance requirements. They are often used in programmable logic controllers (PLCs) and may be used in proprietary communications networks where data rates are 20 Kbits/second or less (the limit for RS-232, but not for later standards).

RS-232 is a physical level protocol. It does not provide higher-level functions. While RS-232 is most often used in asynchronous applications, nothing prevents it from being used for synchronous communications.

3.1.1.2 RS-449, RS-422, and RS-423 Standards

In 1977, to accommodate advances in modems and a need for better electrical performance, EIA published the RS-449/CCITT V.24 and X.24, the RS-422/X.27 (line driver), and the RS-423/X.26 (line driver) standards (Bertine 1980). These are physical-level protocols similar to RS-232,[6] but they extend the operational range of RS-232 to longer cables and data rates as high as 2 Mbits/second. RS-449 replaces the familiar 25-pin D connector of RS-232 with a 37-pin D connector that accommodates 12

[5] Using modern protocols, the traditional link can multiplex several channels on one link, but this was not early practice.

[6] In fact, RS-449 combined with RS-423 is interoperable with RS-232 using an adapter.

additional “interchange” circuits. As well as the actual serial data exchange circuits, interchange circuits include auxiliary circuits to control advanced modems (DCEs) and automatic calling equipment (ACE). As a matter of interest, the widespread use of microprocessors in modems and automatic calling equipment (in 1992) has probably reduced the need for auxiliary circuits in RS-449, since intelligent equipment can make more efficient use of interconnecting wiring.

RS-449 has extended the application domain of RS-232 to communication links of up to 4,000 feet and 2 Mbits/second. This is because the two line-driver standards—RS-422 differential drivers and RS-423 single-ended drivers—have much better performance than the old RS-232 standard. Integrated circuits supporting RS-449, RS-422, and RS-423 design became available even more rapidly than did RS-232, bringing communication links using these standards into the low-cost, medium-performance regime. The CCITT versions of RS-449, RS-422, and RS-423 are specified for CCITT point-to-point use in public data networks.

The RS-449, RS-422, and RS-423 standards are physical level protocols. They do not provide higher-level functions. They may be used for both synchronous and asynchronous communications.

3.1.1.3 DDCMP

DDCMP (Digital Data Communication Message Protocol) is an obsolete byte-count-oriented data communication protocol originated by Digital Equipment Corporation (McNamara 1977, Conard 1980). It can be used on asynchronous or synchronous links and is mentioned here because it provides an example of higher-level protocols that could be encountered in proprietary or obsolete communication systems using RS-232 or RS-449 serial links.

DDCMP attempts to solve the problems of framing, byte synchronization, and data transparency by attaching preambles to data blocks that contain a count of bytes in the following data block. Assuming that the preamble is received correctly, the rest of the frame (the data block) can then be counted as it is received.

3.1.2 Synchronous Character Serial

Synchronous character-oriented serial links require a synchronizing period to lock the receiver clock to the transmitter, after which there are no extra bits included for start/stop operation. Synchronous serial circuits cannot stop and start between characters.

3.1.2.1 BISYNC

The IBM BISYNC (also called BSC) protocol is an obsolete example of a higher-level protocol designed for use on synchronous, character-oriented data links (McNamara 1977, Conard 1980). BISYNC does not use byte counting, but instead uses special link control characters to format, control, and terminate messages. The link control characters can also appear as valid binary data. BISYNC attempts to solve the same framing, byte synchronization, and data transparency problems as DDCMP, with approximately the same success and reliability. BISYNC is mentioned here because similar protocols may be encountered in cost-sensitive proprietary communication links.

3.2 ISO HDLC

Framing, synchronization, and transparency problems with traditional serial standards led to development of the so-called “bit-stuffing” protocols during the 1970s, of which HDLC, IBM SDLC, ADCCP, LAP, and LAPB (all defined in the glossary) are examples (Carlson 1980). The root of the problem is having non-transparent link control characters mixed in with—and indistinguishable from—data. Instead of characters, these frame-oriented protocols transmit variable-length sequences of bits demarcated by uncounterfeitable frame marker bit subsequences. Frame markers typically are a sequence of six one-bits (e.g., 01111110), and no valid data sequence is allowed to have more than five consecutive one-bits. This is accomplished by having the hardware serializer (or deserializer) “stuff” (or “unstuff”) a zero-bit after any five one-bits in transmitted (or received) data. The frame is therefore started and ended by frame markers, and bytes are synchronized at the beginning of each frame. In transmission and reception, hardware performs marker generation (or stripping) and bit stuffing (or unstuffing), and the frame is delivered to software protocol layers synchronized and unsullied by non-transparent link-control characters.

Besides the frame markers, HDLC-like protocols also define a frame format. Figure 9 shows the frame format for HDLC. Integrated circuits have been developed that send and receive HDLC frames, including hardware generation of the CRC frame-check sequence appended to each frame. Most modern high-speed, reliable serial designs use HDLC or an HDLC-like protocol at the frame level. HDLC itself is specified for CCITT’s X.25 protocol.

3.3 Virtual Circuit Protocols

The model for the traditional telephone circuit is the “switched circuit,” which means that for the duration of a conversation a connection is made (switched) between two parties and the entire bandwidth of the connection is available to the parties. In the early days of telephony, this meant that a real wire pair connected the parties through a series of rotary switches. Today it means that a 3-KHz analog communication channel is dedicated to the parties, usually multiplexed with many other analog channels.

A “virtual circuit” is another way of connecting parties by multiplexing data packets instead of channels. Each virtual circuit is identified by control data included in packet preambles, and packets are routed based upon the virtual circuit to which they belong. This is particularly effective for “bursty” data traffic and is a much more effective user of total link bandwidth in these circumstances than is the switched circuit.

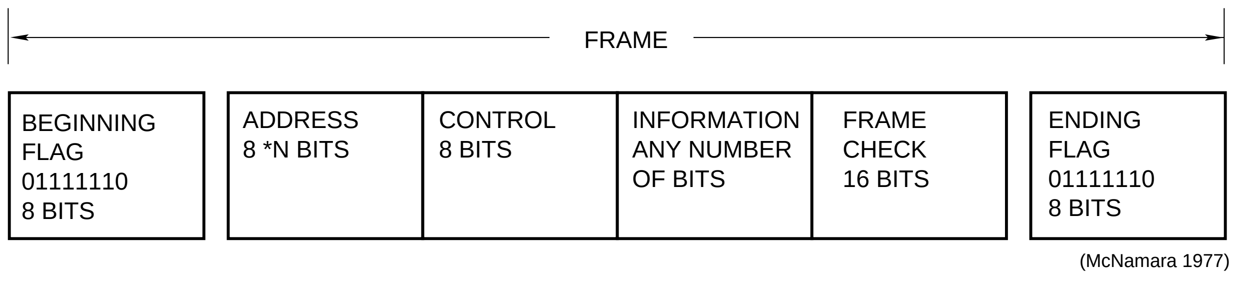

FRAME
BEGINNING FLAG 01111110 8 BITS
ADDRESS 8 *N BITS
CONTROL 8 BITS
INFORMATION ANY NUMBER OF BITS
FRAME CHECK 16 BITS
ENDING FLAG 01111110 8 BITS
(McNamara 1977)

Figure 9. Frame format for HDLC.

3.3.1 CCITT X.25

The CCITT, in its role as the international standards organization for post, telephone, and telegraph interests, has published recommendations for a virtual circuit protocol called X.25 (Folts 1980, Rybczynski 1980). This protocol includes packet formats and procedures for calling (making connection), exchanging data, and call knockdown (terminating connection) on virtual circuits. The procedures are almost a complete analog of making a call on a regular telephone circuit, although datagram service[7] is an exception to this rule.

X.25 is used extensively in public packet-switched data networks where data common carriers[8] provide point-to-point data carriage service. One of its major strengths is that it is a standard upon which the international communications community has agreed.

3.4 ARPANET—TCP/UDP/IP

During the development of the ARPANET, the first large-scale example of a computer network in the United States, a number of higher-level communications protocols were developed that fell into the public domain. Later, computer workstation manufacturers chose to use these protocols because they represented a free, *de facto* standard.

The ARPANET protocols are important because they are often encountered in UNIX-style[9] workstations, and these workstations are increasingly being used in industrial applications. One particular example is a proposed alarm display system for use in nuclear power plants (Carrera and Easter 1991).

All three of the protocols to be discussed work well in office environments and with networks based upon a variety of media and low-level link protocols, including Ethernet, serial lines, and fiber optics. No reports have been found concerning use in Class 1E safety systems.

3.4.1 Internet Protocol

The Internet Protocol (IP) provides routing and addressing between networks (DTIC a, Postel et al 1981). An addressing scheme is defined and under the control of an internationally designated supervisor. The protocol defines preamble formats and encapsulation of data destined for higher-level protocols. The IP preamble contains a selector variable, which designates the next-higher protocol in the protocol stack to use.

3.4.2 User Datagram Protocol

The User Datagram Protocol (UDP) (DTIC c) is an unacknowledged datagram service that rides on top of IP. This protocol defines preamble formats and a method of multiplexing datagrams to recipient tasks within a host. Each datagram is delivered on a best-effort basis, and there is no end-to-end assurance of delivery.

3.4.3 Transmission Control Protocol

The Transmission Control Protocol (TCP) provides reliable byte-stream delivery from task to task (DTIC b, Postel 1979). TCP uses the services of IP to deliver its packets. This protocol performs reassembly, duplicate deletion, and reordering, but has only rudimentary flow control. When used with

---

7 A digital analog of the telegram. See Glossary for further details.
8 An analog of public trucking companies, where network companies provide an electronic delivery service for data.
9 UNIX is an operating system and is also a trademark of the American Telephone and Telegraph Corporation.

IP, it does not conform to the ISO OSI model. Use of TCP has not been reported in safety-critical systems.

3.5 IEEE 802

The development of the Ethernet (Metcalf and Boggs 1976, Xerox et al 1980) sparked enormous interest in local area networks (LANs)[10] that culminated in the formation of IEEE Project 802, a standardization effort. As events unfolded, there were three major physical-level contenders for standardization, and the impasse was broken by standardizing all three. Because of the trichotomy, the 802 committee separated the 802 standards into two sublayers, the Logical Link Control (LLC) sublayer and the Media Access Control (MAC) sublayer. The MAC sublayer concerns all of the media-dependent issues, including specification of the physical layer. LLC deals with media-independent issues and the interface with the ISO OSI network layer. IEEE 802 itself conforms to the OSI physical and link-level layer definitions. The IEEE 802 standard has been adopted by ISO under the names ISO 8802/x, where x matches the 802.x decimal designations.

3.5.1 Logical Link Control

IEEE 802.2 defines the LLC sublayer that is common to all four MAC sublayers described in the following four sections. The LLC protocol sublayer is concerned with four areas: the interface to the ISO network layer above, the interface to MAC sublayers below, service types and procedures to be provided, and a definition of the Protocol Data Unit (PDU) used to encapsulate information handled by this layer. IEEE 802.2 LLC provides the following types of service to the network layer:

- Type 1—unacknowledged connectionless service.
- Type 2—connection-oriented service.
- Type 3—acknowledged connectionless service.

In unacknowledged connectionless service, a sending entity is notified only that a datagram has successfully been transmitted on the medium but not that it was successfully received. No history of transactions is maintained. In connection-oriented service, a logical connection is established between link correspondents, and control of information traffic is maintained using a modulo 8 PDU numbering scheme to keep a limited history of transactions. This provides reliable flow control, sequencing, and error recovery. Type 3, acknowledged connectionless service, is the same as Type 1 service except that the sending entity is notified of success only when the receiver acknowledges by return PDU. Of the three types of service, Types 1 and 3 are used more often for industrial control applications; in particular, MAP 3.0 (to be discussed) specifies Types 1 and 3 LLC.

3.5.2 Ethernet

IEEE 802.3 is the IEEE standardization of the earlier, proprietary version of Ethernet (Metcalf and Boggs 1976, Xerox et al 1980). This MAC sublayer specification defines the physical characteristics of several acceptable transmission media, the access control procedures by which data is transmitted on the media, and the frame format by which PDUs are transmitted. The access method is called Carrier Sense Multiple Access with Collision Detection (CSMA/CD), and is generally considered a non-deterministic communication scheme. Ethernet is used extensively in local area office networks and is the link layer choice for the TOP protocol mentioned in Section 3.6. Lightly loaded Ethernets can be used for control purposes, but the overwhelming majority of control usage is IEEE 802.4, 802.5, or

[10] Typically no more than 1 or 2 kilometers in length.

proprietary networks. Ethernets have also been reported with fiber-optic rather than conductive media, but this represents only a small percentage of current usage.

3.5.3 Token Bus

A deterministic form of CSMA/CD network is defined by the IEEE 802.4 MAC sublayer protocol, which uses a "token" scheme (discussed in the next section) for deterministic arbitration of access to the communication media. This method, appropriately known as "token bus," is the choice for link-level communication in the MAP 3.0 protocol, and is used in MAP and in several proprietary protocols for factory automation. The medium is usually conductive (i.e., copper).

3.5.4 Token Ring

IEEE 802.5 defines a ring bus format that is more appropriate for fiber-optic transmission media than either IEEE 802.3 or 802.4. Stations (i.e., nodes) are connected in a ring topology and a special frame known as a "token" circulates around the ring. Any station having the token is free to transmit, while others are constrained to wait until the token passes to them. The protocol defines rules for creating a token, using the token, relinquishing the token, and the frame format. Token ring is a deterministic communication method, and, although many installations use conductive media, it is useful in situations where the advantages of fiber-optic media are required.

3.5.5 Metropolitan Area Network

The IEEE P802 committee was originally chartered to standardize networks up to 20 Mbits/second, while ANSI's Accredited Standards Committee X3T9 was given responsibility for data rates above this figure. Subsequently, the IEEE committee proposed the IEEE 802.6 Metropolitan Area Network (MAN) MAC standard for interconnecting short-range LANs over distances of up to 50 kilometers with a data rate of 50 Mbits/second (Mollenauer 1988). This standard requires fiber-optic media in a dual-bus (or "slotted ring") arrangement and uses an access method called Distributed Queued Dual Bus (DQDB). The LLC sublayer used is IEEE 802.2. A puzzling question is why the IEEE 802.6 standard was developed at all, considering its similarity to the more commonly used 100-Mbit/second FDDI standard (discussed in Section 3.7). Some part of the explanation may lie in the fact that 802.6 has considerable support among profit-making public data carriers, while FDDI is receiving more support from computer system makers. One thing is clear: IEEE 802.6 MAN has little reported usage in control systems or industrial networks.

3.6 MAP/TOP

In 1980, disturbed by the difficulty of connecting disparate shop-floor automation devices together so that they would cooperate, the General Motors Corporation formed an internal group called the Manufacturing Automation Protocol (MAP) Task Force (GM 1984). GM was learning the lesson that compatible electronic connections did not necessarily mean that communication could take place. Also, real-time shop floor communications were not served well by developments taking place in workstation communications. Simultaneous with General Motors' efforts in shop floor communications protocols, the Boeing Corporation was developing Technical Office Protocols (TOP) to support their database-oriented computer-aided design and engineering activities for designing jetliners (Beeby 1983, Boeing 1985). By 1985, Boeing and GM had joined together under the framework provided by the ISO OSI reference model to produce the MAP/TOP protocols. Figure 10 shows the ISO roadmap for MAP and Figure 11 shows the roadmap for TOP.

The application domain for TOP is the database-oriented technical design arena that includes many disparate computer systems and software technical design tools. The TOP protocols are intended to ease the high-level integration of software and computer products obtained from many sources. TOP uses 802.3 (Ethernet) for link-level data transfer.

The application domain for MAP is Computer Integrated Manufacturing (CIM) (McGuffin et al 1988), which encompasses a range of activities from enterprise administration, computer-aided product design, down to manufacturing cell control. Concerns about real-time performance have led to the definition of a limited form of the MAP protocols called Enhanced Performance Architecture (EPA) or MiniMAP. MAP uses 802.4 (token bus) for link-level data transfer. There are reports of MAP performance in real-time applications (Strayer and Weaver 1988, Mortazavi 1990), but it is typically not used in applications where failure would have disastrous consequences.

3.7 FDDI

FDDI is a token ring optical fiber standard intended for networks of high-performance computers, as an interconnecting network between lower performance LANs such as Ethernet, and as a connection to public data networks through private branch exchanges (PBXs) (Ross 1986, Burr 1986, Ross 1989). The nominal data rate on FDDI is 100 Mbits/second, although this is unlikely to be achieved between any two nodes unless the node processors are fast enough to keep up. FDDI is a MAC sublayer protocol, and most architectures show it alongside and equivalent to IEEE 802.3, 802.4, 802.5, and 802.6 as in Figure 12 (Clapp 1991). FDDI is designed to interface to the IEEE 802.2 LLC sublayer.

There are two kinds of nodes on FDDI networks: single-attachment and double-attachment. Double-attachment nodes connect to two counter-rotating[11] fiber-optic rings that are intended for fault-tolerant failure recovery by automatic ring reconfiguration. Typically, only one ring is used for data traffic and the other is intended for reconfiguration to bypass a failed node. Single-attachment nodes are connected to one fiber-optic ring through the intercession of a special double-attachment node called a concentrator.

Although FDDI means Fiber Distributed Data Interface, copper conductor shielded and unshielded twisted pairs are now included in the standard, because many buildings have existing twisted pair wiring and economic considerations dictate including existing plant if possible.

[11] If nodes are numbered 1,2,3,..., then counter-rotating frames visit nodes in the order ...3,2,1,N,N-1,...

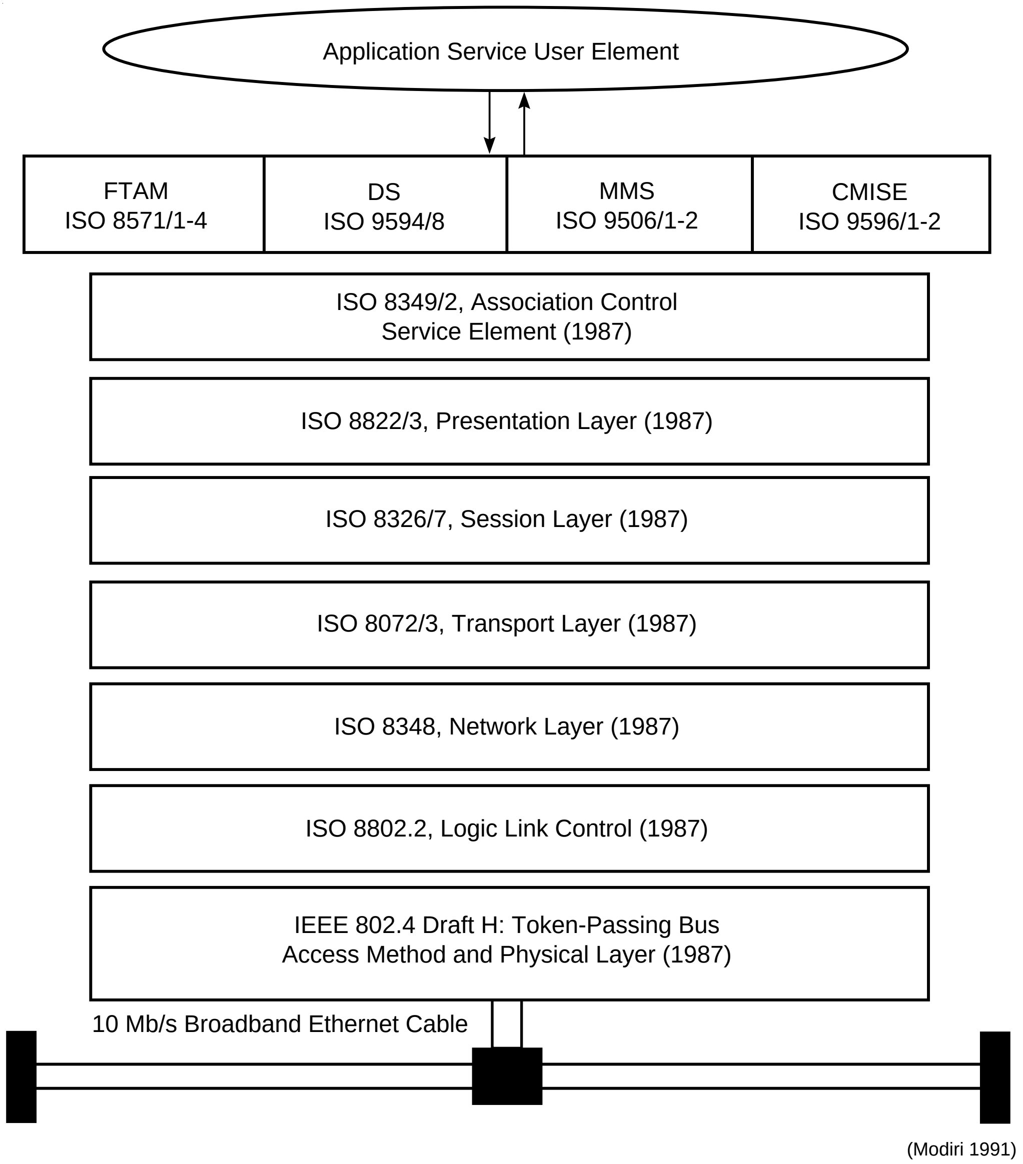
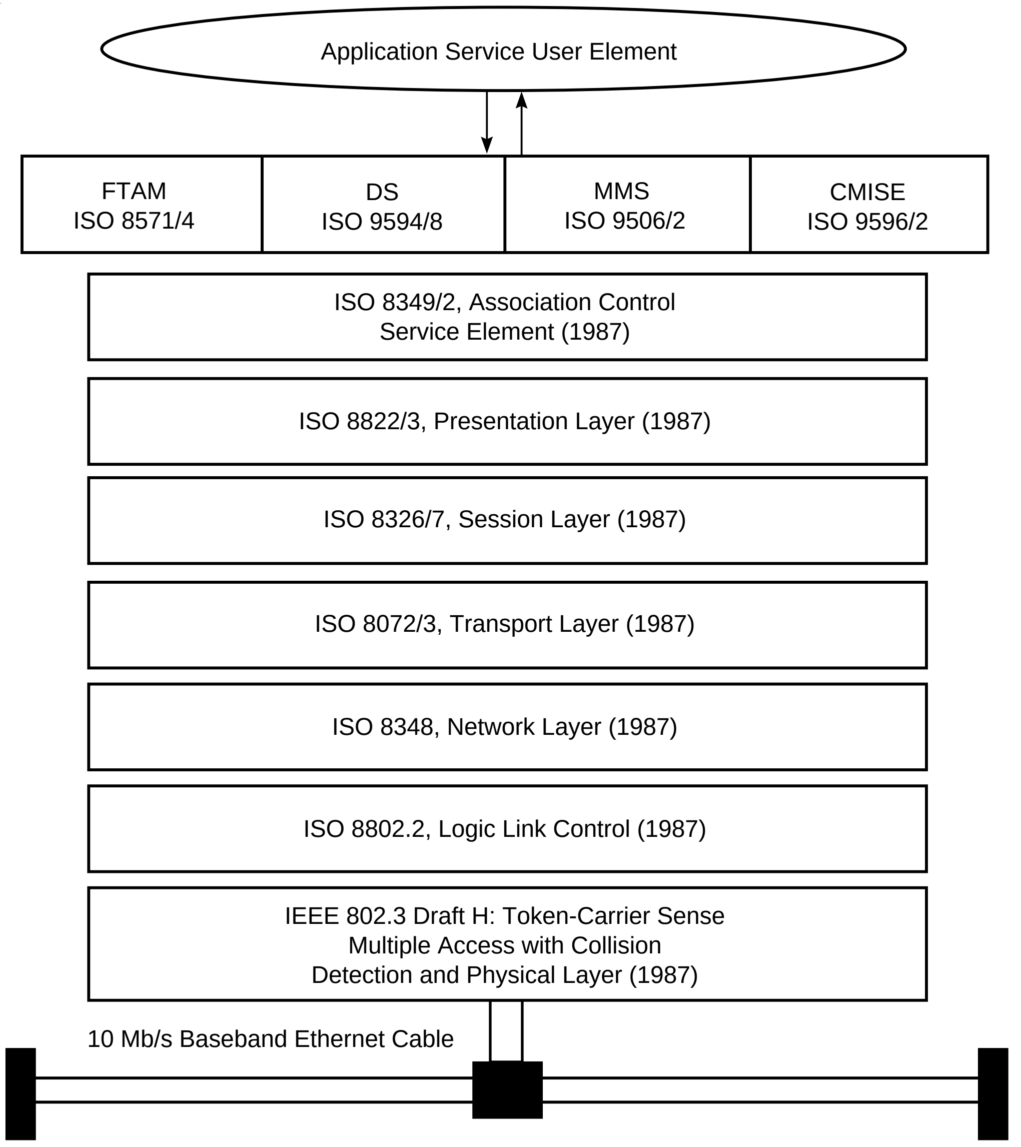

Application Service User Element
FTAM
ISO 8571/1-4
DS
ISO 9594/8
MMS
ISO 9506/1-2
CMISE
ISO 9596/1-2
ISO 8349/2, Association Control Service Element (1987)
ISO 8822/3, Presentation Layer (1987)
ISO 8326/7, Session Layer (1987)
ISO 8072/3, Transport Layer (1987)
ISO 8348, Network Layer (1987)
ISO 8802.2, Logic Link Control (1987)
IEEE 802.4 Draft H: Token-Passing Bus Access Method and Physical Layer (1987)
10 Mb/s Broadband Ethernet Cable
(Modiri 1991)

Figure 10. ISO roadmap for MAP.



Application Service User Element
FTAM
ISO 8571/4
DS
ISO 9594/8
MMS
ISO 9506/2
CMISE
ISO 9596/2
ISO 8349/2, Association Control Service Element (1987)
ISO 8822/3, Presentation Layer (1987)
ISO 8326/7, Session Layer (1987)
ISO 8072/3, Transport Layer (1987)
ISO 8348, Network Layer (1987)
ISO 8802.2, Logic Link Control (1987)
IEEE 802.3 Draft H: Token-Carrier Sense Multiple Access with Collision Detection and Physical Layer (1987)
10 Mb/s Baseband Ethernet Cable

(Modiri 1991)

Figure 11. ISO roadmap for TOP.

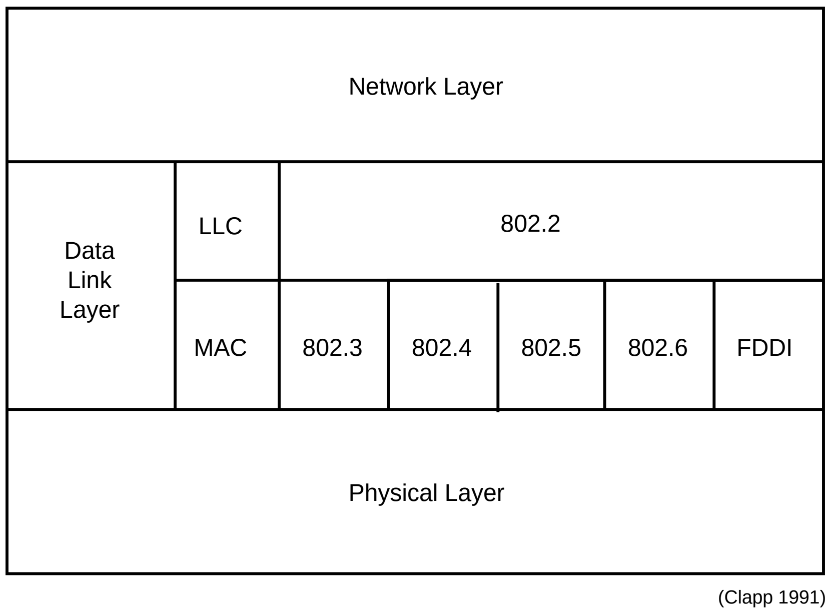


Figure 12. FDDI is a MAC sublayer protocol.

In the future, FDDI rings will be able to operate in two modes: asynchronous frame mode (FDDI-I) and isochronous circuit switched channel mode (FDDI-II). In the first mode (the current standard), the entire bandwidth of the ring is available for typical token-passing operation. In the second mode (an emerging standard), a "cycle master" controls the circulation of a special frame called a "cycle" at precisely timed intervals. A cycle has up to 16 slots for data for 16 different channels, which, given the precise timing requirement, is a time-division multiplexing scheme. The full capability in this mode is sixteen 6.144-Mbits/second isochronous channels, each of which can be subdivided by 3 or 4 into 2.048-Mbits/second or 1.536-Mbits/second subchannels. The subchannel bandwidths match European and American telephony practice respectively, and the American subchannel fits comfortably into a T1 (1.544-Mbits/second) or a CCITT H11 (1.536-Mbits/second) leased line.

While it is certainly possible to use FDDI networks in real-time, safety-critical systems (given enough determination), the intended application domain includes high-performance computer networks, backbone interconnect networks for slower LANs, and connections to ISDN public data networks. Offerings of FDDI network controllers are available in VME bus and Multibus formats, and there is protocol support for TCP/IP and X.25. Given FDDI's adherence to the OSI model, other high-level protocols should easily be supported.

3.8 Field Buses

A field bus is a data communications network that connects sensors and actuators to an intelligent controller or a computer (Pleinevaux and Decotignie 1988). These networks are usually designed for low cost, simplicity, and ease of connection. However, they are often aimed toward specific application areas such as industrial instrumentation, automobiles, or military aircraft, resulting in a fairly wide range of reliability and function. Some current field buses are described below.

3.8.1 PHOEBUS

This field bus was designed for CNC machine tool applications, and the described application was a connection between a PLC, sensors, and machine tool actuators. Peak data rate was reported to be 375 Kbits/second. No maximum length was given, but it is presumably about 4,000 feet, given the data rate and the fact that twisted pair cable was used.

3.8.2 MIL-STD 1553

MIL-STD 1553 is a master–slave[12] twisted pair cable communication bus with a peak data rate of 1 Mbit/second and a maximum of 31 stations. This bus is intended for military aircraft applications as a method of reducing wiring weight and providing light-weight redundancy to compensate for the effects of weapons hits. Maximum length may be limited to aircraft-sized applications. Noise immunity is good, since the bus must withstand military EMI environments. Mil-spec integrated circuits are available for implementing bus interfaces.

There are current efforts to upgrade MIL-STD 1553 to achieve a higher data rate (10 Mbits/second) and make use of fiber-optic media. If the effort succeeds, noise immunity, isolation, length, and data rate should be well within criteria for safety-critical communication systems.

3.8.3 Factory Information Protocol

Factory information protocol (FIP) is a French field bus running at 3 Mbits/second peak. It is OSI compliant (Levels 1, 2, and 7, like MiniMap) and uses a central bus arbitrator to regulate bus activity.

3.8.4 BITBUS

BITBUS is an Intel Corporation-originated communication protocol using an HDLC-like framing scheme at speeds of 64 Kbits/second, 375 Kbits/second, or 2.4 Mbits/second on shielded, twisted pair cable. Up to 250 stations are possible. Like MIL-STD 1553, this bus also has a master–slave arrangement. BITBUS is intended for distributed control applications.

3.8.5 PROFIBUS

PROFIBUS is a German field bus using asynchronous character transmission (8 bits + parity) at speeds between 9.6 Kbits/second to 500 Kbits/second. A maximum of 127 nodes is permitted, although only 32 are permitted per bus segment. It uses a modified master–slave arrangement, with masters exchanging the right to control the bus by passing a token.

3.8.6 IEEE 488

IEEE 488 is an instrument control bus originated by Hewlett-Packard Corporation as a way to connect computers to the instruments the company sold. The bus became so popular that the IEEE standardized it. It is the only parallel bus of the buses mentioned in this paper. It is typically used in localized collections of instruments (within 50 feet) for quick-set-up industrial quality control or test applications. It has some similarities to the Small Computer Systems Interface (SCSI) bus, which is used to connect disk drives and other peripheral equipment to computers. In fact, HP has used IEEE 488 in this way in several of the computer systems they sell.

12 One node (the master) controls the traffic on the network by polling the other nodes in round-robin fashion.

3.9 Proprietary Protocols

The number and purposes of existing proprietary protocols are so extensive that it would be a waste of time to attempt to review even 10% of them. Accordingly, only networks used in industrial automation are discussed here, and those not exhaustively. Each automation vendor, it seems, has one or more proprietary communication networks or has renamed standard networks with a proprietary name. For example, Measurex has DataFreeway, Fisher has Provox, Honeywell has TDC3000, and Foxboro has Foxnet, all trademarked communications networks. Among computer control system vendors and PLC vendors, however, there is a sort of logical hierarchy of network types.

3.9.1 High-Level Proprietary

Typically, automation equipment vendors supply three levels of communication network: a high-capacity broadband network, a mid-level interconnect network, and a lower-speed multidrop[13] network for connecting local controllers with a local supervisory controller. The high-capacity broadband network can be proprietary, but often is a renaming of an existing standard. Ethernet (IEEE 802.3) and MAP (IEEE 802.4) are popular. Even if the high-capacity network is non-standard, such as Measurex's DataFreeway, gateways to standard high-speed networks are usually provided.

3.9.2 Mid-Level Proprietary

Mid-level networks are most often based on high-quality shielded twisted pair cable or multiconductor cable, but some use coaxial cable or fiber optics. Data rates are usually in the range of 19.2 Kbits/second to 1 Mbit/second. Token-passing is typically used as the access control method, but master–slave polling may be found. Allen-Bradley's Data Highway II and Modicon's Modbus Plus are examples. Protocol details differ from one example to the next and are irrelevant to the purposes of this paper.

3.9.3 Low-Level Proprietary

Local controller or PLC connection networks are designed to minimize expense. Consequently, the lowest level in the proprietary hierarchy typically uses twisted pair cable, RS-232 or RS-422/RS-485[14] electrical interfaces, point-to-point or multidrop connection, and a proprietary protocol using some sort of master–slave or "floating" master–slave access method. The latter method simply selects a master node at startup or reconfiguration, after which the master polls the slave nodes as usual. Modicon's Modbus, Allen-Bradley's Data Highway, Siemens' Sinec L1, and Texas Instruments' TIWAY are examples. Data rates are low to moderate (9.6 to 57.6 Kbits/second). Modbus and Data Highway have achieved such market penetration that other manufacturers are often obliged to support these *de facto* standards.

3.10 Subtle Errors in Protocols

A variety of subtle difficulties were encountered in the ARPANET (Kleinrock 1978), the first large-scale computer network to be built. A major problem with more complex protocols is deadlock, and deadlock was found in the ARPANET in at least four different variations. Deadlock is the condition where two or more processes (computer programs) each wait for the others to complete an action for

[13] A multidrop network is a bus with nodes connected at intervals to the network medium. Each such connection is called a "drop."
[14] RS-485 is an enhanced form of RS-422 differential driver suitable for multidrop operation.

which the others in turn are also awaiting. Most often the action awaited is the release of a "resource,"[15] which in the ARPANET's case was usually memory to store or reassemble messages for processing. Other problems included performance degradation due to "looping" (messages traveling in circles), "turbulence"[16] or "thrashing" (swapping tasks to disk so frequently that more time is spent in disk accesses than in doing useful work), and "phasing" (suboptimal resource allocation). Most of these things are still concerns when designing or implementing a new protocol today.

More recently, telephone service over large areas of the United States was disrupted by the "3-bit bug," an error in packet-switching protocols that run in telephone network computers (Watson 1992). A software error caused network routing computers running the software to fail during procedures designed to recover from congestion.[17] Load was transferred to similar routing computers that failed in the same way, resulting in cascading failures that paralyzed the telephone system. Subsequent investigation has uncovered additional vulnerabilities that will probably require protocol changes to repair.

A dismaying aspect of the history of protocol errors is that with a few exceptions, the causes cannot be neatly categorized. One important exception, automatically generated messages, is discussed next.

3.10.1 Chain Reactions

Error recovery is always an area of serious concern in protocols. A seemingly reasonable response to an error is to generate a message to the initiator of the message causing the error or to forward messages to other entities that may respond to the error. If the returned or forwarded error message also generates an error, an infinite or very large finite chain reaction is begun which may overload the network to the point of failure. Such chain reactions have been reported in Ethernet broadcast messages, electronic mail forwarding loops, "flood" routing protocols, and query forwarding schemes (Manber 1990).

A feature common to all these failures is the automatic generation of additional messages. The author (Manber) suggests that any such scheme should be scrutinized closely.

4. Discussion

4.1 Nuclear Reactor Communication Networks

Data communications networks are being proposed to replace conventional wiring in nuclear power plants. Reasons for doing so include greater reliability, increased function, more appropriate connections to computers, and decreased cost. These goals can be achieved if the network implementation is good and the application is appropriate.

For the purposes of this paper, there are four application domains for data communication networks in nuclear power plants:

- Class 1E safety systems.

[15] A resource is a buffer (memory area), a device (line interface, printer, disk), or CPU time required to accomplish a task.

[16] Turbulence in the ARPANET is defined as excessive retransmission of packets because work was discarded before completion due to out-of-order packet transmission.

[17] Congestion is the term used to describe the situation of too many messages for the memory to handle and not enough CPU power available to process them. Another descriptive term is "overload." In distributed networks, congestion can occur locally at "bottlenecks" while not occurring generally in the network.

- Safety monitoring systems.
- Reactor control systems including process parameter display systems.
- Everything else.

The most stringent performance and reliability requirements apply to Class 1E systems, while no regulatory requirements apply to the fourth domain. Networks in the four domains can be and are connected together, but the differences in safety requirements mean that connections to less-safe networks must not impede or affect the operation of more-safe communication networks.

Class 1E safety systems include the Reactor Protective System (RPS), the Engineered Safety Features (ESF), the Engineered Safety Features Actuation System (ESFAS), the Neutron Monitoring System (NMS), some portions of the Reactivity Control System (RCS), and possibly some portions of the plant annunciator system. The safety monitoring system displays variables important to safety, and some sensors and display equipment that it uses may be Class 1E depending upon the use of the variables for manual safety actions, release monitoring, or confirmation of safety function. The reactor control system includes all non-Class 1E control equipment that is used to operate the reactor and may include the balance of the plant annunciator system. "Everything else" covers balance-of-plant control systems, business functions, economic dispatch, maintenance, and other administrative functions that may be accomplished with computers and for which there are local area computer networks. Only the first three domains will be considered in the sequel.

4.1.1 Description of Domains

Nuclear power plants are stressful environments. Reliable data communication networks have to withstand challenges from three areas:

- Nuclear and electrical radiation.
- Electrical and nuclear faults and consequent sequelae.
- Human (fallible) operation and maintenance.

To survive in this environment, desirable qualities for both Class 1E systems and reactor control system data communication systems include:

- Physical robustness.
- Noise immunity.
- Isolation.
- Fault tolerance.
- Maintainability.

In addition, any practical system must perform well enough to support the purpose for which it is intended. Besides a need for enhanced reliability, lower performance requirements may be the major difference between data communications networks intended for Class 1E applications and those intended for control system applications.

4.1.1.1 Class 1E Safety Systems

Reactor protection systems typically consist of groups of sensors or instruments connected to processing and decision elements, which are in turn connected to safety-related actuators. Traditionally, these connections were made with one circuit per sensor or actuator, and both wiring and sensor were replicated when redundancy was required. The processing and decision elements performed relatively simple calculations (usually analog), and protective action was immediate when needed.

Since safety systems terminate unsafe conditions and initiate mitigating actions, there was little need for continuous Proportional-Integral-Derivative (PID) feedback control in reactor protection systems.[18]

In some recent plans for computerizing reactor protection systems, vendors propose to replace individual circuit wiring with data communication networks and to replace analog processing and decision elements with computers. A significant feature of these proposals is the multiplexing of sensor and actuator circuits through the data communication networks.

At least in existing nuclear power plants and in evolutionary designs, the required reactor protection network data rates are moderate and signal processing and decision algorithms are relatively simple and fast. This reflects the facts that safety systems are usually simple (for reliability) and that safety actuating equipment (control rods, core cooling, and containment isolation) is slow on an electronic time scale. Because of the simplicity, low speed, and well-defined nature of reactor protection systems, they are good candidates for state model design.

4.1.1.2 Reactor Control Systems

Reactor control systems, while technically not Class 1E equipment, can still have significant effects on reactor safety if they cause challenges to reactor protection systems. These challenges can occur three ways:

- Misoperation, directly challenging the protection system.
- Failure to operate, allowing reactor to drift to an operational limit.
- Failure to control an abnormal excursion.

Control system design has similarities to protection system design. The control system has sensors, actuators, signal processors, and control algorithm processors arranged in much the same way as protection systems use sensors, actuators, signal processors, and decision processors. Data communication networks can be used to connect these elements in a similar manner to that used in reactor protection systems. However, reactor control systems have four additional factors that complicate the task of selecting a reliable data communications system:

- There is much more human interaction.
- PID or state estimator-based feedback control is often used.
- Control algorithms and architectures are more complex.
- Much more data is handled.

The two middle factors may mean that timing requirements on control system communications are stringent because too much delay will cause the system to become unstable. The first factor nowadays results in considerably more required processing power and data movement because of graphical user interfaces and data display, an element much diminished in reactor protection systems. Finally, the variety and quantity of data used by reactor control systems may be at least an order of magnitude greater than that handled by reactor protection systems. Typically, all of the data produced by the reactor protection system is either duplicated or passed through to the safety monitoring, reactor control, and process parameter display systems, to which is added data and alarms produced by the control system.

4.1.2 Key Questions To Ask

Although there are many examples of system failure due to misunderstanding of crucial physics, many other systems fail to perform satisfactorily because the designers do not understand what is wanted or needed. In other words, the designers understood the physics, but had not considered the

[18] There are often binary feedbacks, such as valve position switches.

system implications. Asking questions can focus attention on the real requirements for data communications subsystems; a systems vendor who cannot answer such questions may be more likely to supply a data communications system that is unsafe, unreliable, or fails to perform adequately in all circumstances.

1. ***Is it an event-based or a state-based system?***

   If there is uncertainty about the model to which the system is being designed, the system's performance under upset or high load may be difficult to predict.

2. ***Is there an accurate picture of the layout?***

   If the layout (detailed architecture) is not known, accurate performance modeling and simulation is impossible. Estimates of data rates can only be approximate because the communication links between nodes and the number of nodes are unknown.

3. ***Are the data rates known between all nodes of the architecture? Are they known for upset and error conditions?***

   If data rates exceed link capability or the ability of nodes to handle traffic, the system will suffer congestion. A node should have sufficient computational capacity left after handling communication traffic to perform its other functions. If a system vendor cannot answer this question, it may be a sign that the vendor has not considered the application very carefully.

4. ***Is the message mix known? Is it known for upset and error conditions?***

   Data rates are usually specified in bulk bytes (or bits) per second. This does not include overhead (extra bits) incurred by various communication schemes, which is usually added to each message. If the message mix consists mainly of short messages, the bulk data rate supported by the links and nodes may not give a true picture of how the system will perform.

5. ***Are the timing and delay requirements known?***

   The data rate and message mix combined with theoretical support can give a reasonable estimate of mean message delay and delay probability distributions for links between each pair of nodes. Does the estimate meet requirements for system performance? More importantly, have system performance requirements been used to derive message timing accuracy and delay requirements between nodes?

6. ***Is the system "deterministic" and are the effects of error recovery accounted for?***

   If the system supplier's safety and reliability projections depend upon deterministic performance of the data communication subsystem, then it is important to know how the system recovers when non-deterministic faults are encountered. All data communication systems are non-deterministic for some subset of system failures. Token-passing networks in particular may incur random delays when reconfiguring or recovering from lost tokens.

7. ***Is the actual link data capacity including interface, operating system, and protocol performance effects being quoted, or is the vendor basing calculations on raw media bandwidth?***

   For many popular data communications networks, node-to-node data rates are rarely close to peak medium bandwidth.

8. ***What are the media requirements?***

   - Isolation?
   - Noise immunity?
   - Physical robustness?
   - Fault tolerance?

- Bandwidth?
- Raw bit error rate?
- Maintainability?

9. Does the design meet independence requirements?

Connections to less-safe networks must not impede or affect the operation of more-safe communication networks, nor should redundant divisions interfere with each other.

10. What are the communications error performance requirements?

- What is the undetected error rate required and how is this determined? Will there be full-scale testing to determine the actual rate?
- What undetected error rates are expected during plant upset when noise sources may be very active?
- What error-detection and correction system is to be used and how much of the system traffic is dedicated to this task?
- What are the worst-case consequences of undetected errors? What can be done to mitigate these consequences?
- What does the system do when it encounters an unrecoverable error?

11. What are the protocol requirements? What services should the protocols provide?

There are now many standard communications protocols. Sometimes they are selected because they are convenient, not because they are appropriate to the problem being solved.

12. Is the medium proprietary? Are the protocols proprietary?

Proprietary media and protocols may have the disadvantages of having less reported experience in the literature and little or no theoretical support in the literature for performance and reliability. There also may be logistic considerations, which affect safety because of insufficient material support and inability to correct design faults discovered later.

13. Is there theoretical support for performance and reliability?

Throughput and delay characteristics of certain communications media and protocols have been calculated and presented in the literature (see, for example, Kleinrock and Tobagi 1975, Tobagi 1980)). The models used and the results obtained can be used to predict performance and also to provide guidance for testing.

14. Is there experimental support for performance and reliability?

Experimental measurement of the performance and reliability of a particular implementation of a data communications system gives greater confidence that the implementation will work as expected. However, experiment supported by theory is preferable to either by itself. Measurements have been reported for certain popular networks (Shoch and Hupp 1980, Strayer and Weaver 1988, Mortazavi 1990).

15. Is there an installed base? If proprietary, how many suppliers support the medium and the protocol software?

A large installed base is a source of performance and reliability statistics as well as a source of user anecdotes. Since there may be several implementations of both communications hardware and software, an installed base is a useful source of comparisons.

16. ***Is there a good match between node processors, network controllers, operating system, and protocol stack?***

This question focuses attention on design balance. Is the design a hodgepodge of conveniently available pieces, or is it a well-thought-out, balanced answer to system requirements?

4.1.3 Design Elements

Certain approaches to a communications system design can make it more difficult to achieve reliable service and, conversely, other approaches are easier to understand, verify, and test. Usually the reasons for using riskier design elements are to increase throughput and reduce message delay or to add desirable features. In these cases, the use of difficult design approaches is an intentional, planned part of the development process. However, sometimes design techniques are used without appreciation for the difficulty involved. It is the latter instance that gives more cause for concern.

The presence of difficult design elements is not *prima facie* cause for design rejection, but it is cause for deeper investigation. The elements listed below should be regarded in that light.

4.1.3.1 More Difficult Safety and Reliability

1. ***Load-correlated load.***

Load-correlated load is additional load on a communication system that is a positive function of system load. A typical example is low-level error-recovery attempts correlated with increased network traffic, such as those that occur in ALOHA-type systems (Kleinrock and Tobagi 1975). Pure ALOHA systems are unstable over 18% load, and experience an almost complete loss of network throughput when offered load exceeds this figure.

2. ***Chain reactions (automatically generated messages).***

Chain reactions are caused by automatically generated messages that circulate between programs running in two or more nodes, usually because each message generates additional error conditions, which in turn produce more messages. Chain reactions have been seen in low-level internet protocols, electronic mail systems, error responses induced by broadcast messages, and network routing messages (ARPANET) (Manber 1990).

3. ***Framing and byte synchronization in software.***

Experience reported in the literature (Carlson 1980) suggests that hardware is the better place to handle framing and byte synchronization because the functions are easily mechanized and doing it in software imposes timing constraints that are unnecessary with the modern equipment available.

4. ***Violations of modularity.***

Modularity limits the propagation of errors and is one of the main points of the ISO OSI model. Non-modular systems have interdependencies that can result in unplanned side effects that are not always benign.

5. ***Sequence dependencies.***

A sequence dependency in a distributed system is a condition where software running in one part of the system depends for its own correct function upon a sequence of actions taken by software running in other parts of the system. A consequence can be deadlock (Kleinrock 1978) that occurs because of sensitivity to certain resource-allocation sequences. Other types of deadlock and unfortunate results are also possible.

6. ***Poorly defined grounding, shielding, and isolation plans.***

Communications media often extend into electrically hazardous portions of a plant. Poorly conceived grounding, shielding, and isolation can admit destructive electrical energy into an otherwise well-designed safety system, defeating its purpose.

7. ***"Dead"-but-present protocol layers.***

A dead protocol layer is a layer in a protocol stack for which software is present but which is not really needed by the application. Messages pass through the layer and the software is executed with generally zero effect. This increases the amount of software executed, increasing processing delays and also the chances of encountering software faults.

8. ***Event-based systems.***

The performance of event-based systems during upsets or overloads is difficult to predict.

9. ***Lack of theoretical support for performance.***

Without a theoretical model for communication system behavior, it is difficult to say that the system is working the way it should. Theoretical models are given for many standard networks (Kleinrock and Tobagi 1975, Tobagi 1980, Tobagi et al 1978). Methods for calculating and measuring low-level performance are described by several papers (Stuck 1983, Tobagi et al 1978, Inose and Saito 1978).

10. ***Position-sensitive data.***

Position-sensitive data is identified by its position in a transmitted frame. This technique is used in many multiplexing schemes with effective results, including most TDM schemes. Good results depend upon careful planning and configuration control, but errors in applying the method are difficult to detect because a receiver has no way of checking data identity.

11. ***No study of error rates.***

Insufficient knowledge of expected and required error rates can result in unpleasant surprises. The effects of unplanned and excessive error rates can range from lower-than-expected throughput to misoperation of the system.

12. ***No study of data rates.***

Depending upon the design, communication systems are subject to congestion and unpredictable delays when exposed to higher-than-expected offered data rates. A system may exhibit "choke points," which are individual overloaded communications links due to their position in the architecture.

13. ***No formal design development process.***

The lack of formal design process makes it difficult to tell why things were done, when they were done, and to what stage the design has progressed. In communications systems, where there often is significant interaction between hardware and software design, it is also difficult to determine if hardware and software design are coordinated.

4.1.3.2 Easier Safety and Reliability

1. ***Modularity.***

Modularity limits the propagation of errors, makes systems easier to modify, and limits the intellectual concerns of the designer to manageable proportions (Parnas 1979).

2. ***Encapsulation.***

   Encapsulation is not only a form of modularity, but permits reliable internet transportation of messages in communication systems.

3. ***Hardware-framed messages.***

   Hardware-framed and synchronized messages have not only been found to be more reliable (Carlson 1980), but concerns of framing and byte synchronization are removed from protocol software, which already has more than enough to do.

4. ***Sequence-independent design.***

   Sequence-independent designs do not experience deadlocks and may require fewer states to describe the system.

5. ***Comprehensive grounding, shielding and isolation plan.***

   Fault propagation is limited at well-defined boundaries in the grounding, shielding, and isolation scheme. Noise performance is guaranteed because of grounding and shielding chosen for the expected EMI environment.

6. ***OSI model standard protocol with minimum layers.***

   The use of the OSI model enforces modularity, separation of function, and interface requirements between protocol layers. Also, some of the protocol layers can be OSI-compliant standard protocols even if other layers are proprietary.

7. ***Theoretical support for claimed performance.***

   The communication system can be tested, and conformity to theoretical predictions increases confidence that the system performs correctly.

8. ***State-based system.***

   Validation of state-based systems is easier because several problems such as deadlock and congestion can be precluded by reachability analysis (see Section 4.2.1.2).

9. ***EMI-immune and inherently isolated media.***

   Propagation of electrical faults into equipment is eliminated, which reduces the chance of common-mode system failure. Induced electrical noise problems are eliminated, which reduces the noise problems to inherent process-generated noise (e.g., thermal and semiconductor noise).

10. ***Position-insensitive data.***

    Data that is identified with tags or other methods, rather than its position in a frame, has added redundancy that can help catch configuration errors.

11. ***Comprehensive study of noise sources with error detection, and correction system based on expected error rate.***

    Error detection algorithms are more sensitive to some kinds of noise than others, depending upon the mathematical characteristics of the algorithm. Understanding the principal types of noise present in the communication system enables a logical choice of an appropriately matched error-detection and correction scheme. A mismatch could result in undetected errors or more overhead than required.

12. ***Comprehensive study of data rates required, including error recovery issues, protocol delays, and message mix.***

    A good data rate study helps avoid unexpected congestion and communication system overloads when the system is subjected to loads under real operating conditions. The best time to size the system is before it is built, not after it has been delivered.

13. Coordinated formal hardware and software design development processes.

Formal design procedures are beneficial for any software project, and communication system design is no exception. Formal design techniques leave organized documentation about why, when, and where things were done that makes understanding the system in the event of difficulties much easier. Testing is also improved because the formal design includes performance objectives that can be measured in the final product.

14. Use of protocol engineering tools for formal specification, validation, and test.

Communication protocols lend themselves well to formal mathematical specifications because they can be described as interacting finite-state machines. Some methods and tools for describing, validating, and testing communication protocols are described in Section 4.2, Reliability, and Section 4.3, Techniques and Tools.

4.1.4 Media Suitability

Because of the great variety of communication rates, application objectives, and media characteristics, there is no easy answer to the question "which medium should be used?" Ethernet (IEEE 802.3) has at least three choices to fit the economic and performance requirements of any particular installation. Power plant installations complicate the issue because electrical faults can be large enough to destroy steel switchgear, and noise from equipment such as high-power thyristor exciters can pervade the plant.

In very hazardous or noisy electrical environments, the medium of choice is fiber-optic cable because of its electrical isolation and noise immunity. The medium itself is not a limitation on the data rates that would be expected in reactor protection or control systems. For example, the 100-Mbit/second data rate of FDDI may well be overkill for reactor protection system applications. Fiber comes in a number of sizes and as "single-mode" or "multi-mode." The mode designations refer to the propagation modes possible in the fiber, with single-mode supporting very high data rates but requiring more precise connectors.

In more benign environments where high data rates (greater than 2 Mbits/second) are required with good noise immunity, coaxial cable is a good choice. Coaxial connectors have fewer difficulties than fiber optic connectors, but are more expensive than the screw terminal connection that is possible with twisted pair cable. Coaxial cable is used in Ethernets, MAP networks (IEEE 802.4), and for various broadband "backbone" network applications designed to connect clusters of computers or controllers together.

In electrical environments where noise and fault propagation are of less concern, twisted pair or shielded twisted pair cable is a good choice for low to moderate data rates (9.6 Kbits/second to 2 Mbits/second) over short to medium distances (10 m to 1 Km) where low cost is important. Low-frequency fiber-optic cable[19] can replace twisted pair cable in applications were isolation is required without a large increase in expense. Twisted pair cable is often used between PLCs and low-level electronic control equipment in distributed process control systems.

All the media mentioned have two problem areas. The first is connectors. A medium rarely gives trouble in its middle, but most often at connections. Therefore, a medium should not be considered separately from the available connectors. The second problem is that the most appropriate medium for an application may not be supported by a desirable communication protocol. In this case, the choices are

---

19 The cable core is larger, the connectors less precise, and the electrical-to-light converters are capable of only a few MHz.

to design a special protocol (not very easy) or to pick an available but somewhat unsatisfactory protocol.

4.1.5 Protocol Suitability

Selection of protocols for use in communications systems in nuclear plants is very much dependent upon balancing safety and reliability against performance (speed and throughput) required for a specific application. Other issues such as modularity, maintainability, and commercial availability are also important. For very reliable systems, which include reactor protection systems, a protocol designed using recent results in protocol engineering (King 1991, Holzmann 1992) may be the only way to obtain sufficient assurance of safety. Reactor control systems, which require more performance but can tolerate lower reliability, may use standard or proprietary protocols developed prior to more general use of Formal Description Techniques (FDTs). In the following, the discussion will be limited to protocols suitable for highly reliable applications.

Use of a standard protocol is not a guarantee of suitability. As has been discovered (Budkowski and Dembinski 1987), the specification of distributed systems is a far more difficult task than specifying sequential single-thread systems. Computer communication protocols are one form of distributed system. Standard and proprietary protocols exist whose specifications have not been validated by formal techniques and which may contain specification errors.

A suitable protocol will contain only the functions required for the application. Certain reliability properties (Section 4.1.5.2) will be present and the protocol specification will be free of errors. Formal method development, formal testing, and field experience will have verified as far as possible that the particular implementation of the selected protocol matches the protocol specification.

4.1.5.1 Abbreviated OSI Model

The ISO Open Systems Interconnect Model is a useful model for dividing a communications task into modular layers with specific functions assigned to each layer. The entire seven-layer model is probably not useful for local area networks used in highly reliable systems. Many of the functions are simply inappropriate where internetworking is not required and simplicity is valued. This fact has been recognized in the MAP protocols (Mortazavi 1990, Strayer and Weaver 1988) with the definition of an abbreviated “miniMAP” protocol for high-performance local applications.

Similar to miniMAP, the retention of OSI model layers 1, 2, and 7 is appropriate for highly secure, highly reliable reactor protection communication systems. Maintaining structural compatibility with the OSI model also has the benefit that protocol engineering tools and techniques developed for that model will be directly usable.

4.1.5.2 Reliability Properties

Certain properties are necessary, but not sufficient to guarantee reliable performance. These are *safety*, *liveness*, and *real-time performance* (King 1991). *Safety* properties describe what a system is allowed to do. *Liveness* properties describe what it must do. *Real-time performance* properties describe how quickly it must do its job to meet externally imposed system deadlines.

4.1.5.3 Freedom from Errors

Cumulative experience in protocol design has identified a number of errors common to many protocol specifications (Kakuda and Saito 1991, King 1991, Probert and Saleh 1991). These include:

- Missing state transitions.[20]

[20] A common model for communication protocols is the Finite State Machine (FSM) (Brand & Zafiropulo 1983).

- Redundant state transitions.
- Non-executable state transitions.
- State ambiguity.
- Deadlocks.
- Livelocks.
- Unspecified receptions.
- Unspecified transmissions.
- Channel overflow.

State transition errors involve under- or over-specification of protocol actions in response to stimuli (such as message reception). State ambiguity is a situation where entities executing the same protocol lose synchronization because several incompatible states may coexist between the entities. Deadlock is (simply described) the situation where protocol entities await each other because each has something the other wants. Livelock is essentially the same situation but with endless rounds of useless negotiation. An unspecified reception is a message the protocol was not expecting and cannot deal with in its current state. An unspecified transmission is either a transmission that should have been sent but was not or one that should not have been sent but was. A channel overflow is a situation where the number of messages offered to a channel by a protocol entity is unbounded or exceeds the channel capacity—in short, a chatterbox or lack of flow control.

Modern protocol engineering techniques have synthesis and validation tools that address these known problems. These will be covered more in Section 4.2, Reliability, and Section 4.3, Techniques and Tools.

4.1.5.4 Field Experience

Field experience can be divided into two areas—testing of a particular implementation of a protocol and actual field experience by users of the protocol. It is important to realize the distinction between validation of a protocol specification and verification that an implementation meets the specification. Validation ensures that the specification has no logical errors; verification ensures that a particular software product that implements the specification actually does so. Besides using formal methods to produce the software product, testing is a complementary method of verification.

Testing of a suitable protocol does not involve anecdotal user reports or the simple application of an electronic network analyzer. Methods are now available that start from the protocol Formal Description (FD) and generate, either manually or automatically, test cases, test executions, behavior analyses, and reports. No protocol implementation intended for reliable service should fail such testing. The current state of testing is discussed in the following section on reliability.

Finally, Formal Description Techniques (FDTs), specification validation, formal method development, and FD-directed testing cannot absolutely ensure communication protocol reliability. Reports of user experience of "reliable" communication protocols should not be ignored. Good user experience cannot "prove" that a protocol is reliable, but derogatory reports are certainly counterexamples.

4.2 Reliability

As discussed in Section 2.6, Data Communications Performance, many factors can affect performance and, by implication, reliability. Failures in any of these areas can affect communication system reliability:

- Medium.
- Interfaces.
- Fit to CPU.
- Operating System.
- Protocol Software.
- Appropriate use of communication services.
- Effects of performance on reliability (e.g., congestion).

The first four areas are either susceptible to conventional reliability improvement measures or are out of the scope of this report. First, they will be discussed briefly, then the last three items will be considered in more detail.

Assuming an appropriate selection of medium, as in Section 4.1.4, Media Suitability, the reliability of media and connectors can be enhanced by normal quality control inspection and testing. Interfaces are electronic equipment whose reliability can be projected by extensive experience in the electronics industry. However, some modern network interfaces contain extensive on-board software that is indistinguishable from any other protocol software. In these cases, the methods described under Section 4.2.1, Protocol Software, cannot be avoided for assurance of highest reliability. Besides the usual concerns of matching interfaces to a CPU backplane bus, node CPUs should provide memory protection, interrupt processing, and DMA capability appropriate to the complexity of the protocol envisioned. The operating system used, if any, must also provide resource management services so that existing CPU capabilities are available to protocol software and network interface drivers as needed. All three of these considerations (*Interfaces, Fit to CPU, Operating System*) can be analyzed and relative priorities balanced using *safety analyses*, *requirements analyses*, and *specification generation* techniques detailed in the literature (Sparkman 1992, Lawrence 1992a).

4.2.1 Protocol Software

In one respect, communications protocols are no different than any other software. The same need exists for ordered and controlled software development methods as for any highly reliable software. These methods have been covered (Sparkman 1992, Lawrence 1992a, Zucconi and Barter 1992) and will not be further discussed here. Because of the very subtle interactions of communicating entities and because protocols are susceptible to formal descriptions, a body of work has grown that is particular to communications protocols. Formal methods were employed in 1980 or earlier (Bochmann and Sunshine 1980) and two international standards bodies (ISO and CCITT) have cooperated to standardize several FDTs (Katsuyama et al 1991). The protocol specification and validation process is now fairly well understood, and recent recommendations propose use of formal methods (Holzmann 1992, see also Williams 1992). In the following, the current art in synthesis, validation, and verification of communication protocols will be outlined.

4.2.1.1 Design of Reliable Protocols

The most general description of a communication protocol is as a collection of communicating parallel programs with no constraints on either the timing of communications or the channel capacity. A more restrictive description that is used in many modern approaches is the Finite State Machine (FSM) description (Brand and Zafiropulo 1983), in which the parallel programs are modeled as state machines connected by communication channels. The essence of design is to synthesize a protocol having desired functions, describe it unambiguously, and prove that the hazards mentioned in earlier sections of this paper do not apply (King 1991, Probert and Saleh 1991, Kakuda and Saito 1991, Holzmann 1992). This is the synthesis, specification, and specification validation problem. The second part of the design problem is to convert the validated protocol specification to a software design. If an FDT has been used for describing the specification, formal methods covered elsewhere (Williams 1992) can be used to verify that the software designed does indeed match the specification. Finally, the third part of the design problem is to test the implemented software by generating test cases that demonstrate its properties and can be analyzed for evidence of errors. Considerable work has gone into standardizing protocol testing methodologies, spearheaded by ISO and CCITT and supported by the national standards agencies of several countries (Rayner 1987, Linn 1989).

The synthesis problem still requires the most subjective human intellectual intervention. This is where human objectives are codified into unambiguous statements of function or behavior. By combining validation and simulation interactively with synthesis, the human designer is allowed to see the implications of his choices. Some tools will be described later that assist in this process.

4.2.1.2 *Validation of Protocol Specifications*

To understand the validation problem, it helps to consider a more restricted view of communicating protocol entities as N communicating Finite State Machines (FSMs) connected by finite-capacity channels. In this scenario, at any instant the system of N machines can be described by a 3-tuple

$$\{ < s_{p1}, s_{q2}, \ldots s_{uN} >, < x_{11}, x_{12}, \ldots x_{NN} >, < f_1(s_{p1}, X), f_2(s_{q2}, X), \ldots f_N(s_{uN}, X) > \}$$

where

$<s_{p1}, s_{q2}, \ldots s_{uN}>$	are the current states of the N FSMs,
x_{ij} Œ X	is the channel content from node i to node j (x_{ii} is empty),
X	is the set of x_{ij}, and
$f_i(s_{pi}, X)$	is the *i*th next-state mapping function.

Since the FSMs are separate and execute in parallel, and there is communication channel delay, progress of the system occurs by interleaved state changes, message transmissions, and message receptions. The validation problem is to ensure that the protocol progresses and has no errors for any possible interleaving. For convenience, the previous list of common errors is repeated below:

- Missing state transitions.
- Redundant state transitions.
- Non-executable state transitions.
- State ambiguity.
- Deadlocks.
- Livelocks.
- Unspecified receptions.
- Unspecified transmissions.
- Channel overflow.

The expansion of interleaved system states and analysis to detect errors is called *reachability analysis* (Lin and Liu 1992). The concept is fairly simple, but the implementation is marred by the *state explosion*[21] problem of large protocols. Various methods are distinguished by the different ways used to reduce the number of states expanded and searched. Also, many implementations of validation systems work only with N = 2, although some are available with more ambitious specifications. Some currently available synthesis and validation systems are discussed later.

[21] The total number of states expanded rapidly exceeds available computer memory.

4.2.1.3 Verification of Protocol Implementations

A validated protocol specification can be used to generate a number of implementations, usually for different computers and interface gear that require different languages, operating systems, and operational sequences. The process of ensuring that a particular implementation matches the specification is called *verification* and is a common feature of high-quality software implementation efforts (Zucconi and Barter 1992). The protocol design community, led in particular by the standards organization ISO, has developed a set of specific test architectures for testing protocol implementations and verifying as far as possible that the implementation conforms to the standard specification (Rayner 1987). The test architectures are oriented toward layered models similar or identical to the ISO OSI model.

The basic concept, illustrated in Figure 13, is that a single layer of a protocol is surrounded by an *upper tester* (UT) and a *lower tester* (LT), both coordinated by *test coordination procedures* (Linn 1989). The testers are software, or combinations of software and hardware, which interface to the Implementation Under Test (IUT) through Points of Control and Observation (PCOs). The testers exchange Protocol Data Units (PDUs) and service requests with the IUT and compare its behavior to that expected of the specification. Figure 13 is known as the *local method* of protocol testing. The local method assumes that both upper and lower interfaces to the layer under test are accessible as PCOs.

It may not be possible to gain access to layer interfaces directly. In this case three other models (Linn 1989) are available to test protocol layers with inaccessible interfaces or which are embedded in several other protocol layers. Figure 14 illustrates the *distributed method*, Figure 15 the *coordinated method*, and Figure 16 the *ferry method*. The *remote method* (Figure 17) is another single layer method.

The test methods just described indicate how to "hook up" a protocol implementation to be tested. The problem of how to drive the test harness, or which test cases to use, is still a subject of active research. Since the number of possible operational sequences for large protocols is effectively infinite, even with modern memory and computational support, efforts have focused either on selecting "relevant" test cases, automating the generation of test cases, or both. Linn (Linn 1989) surveys several historical methods (some still in use):

- Distinguishing sequence (DS) method.
- W method.
- Transition tour (T) method.
- Unique input/output (UIO) sequences method.
- The Rural Chinese Postman (RCP) method.

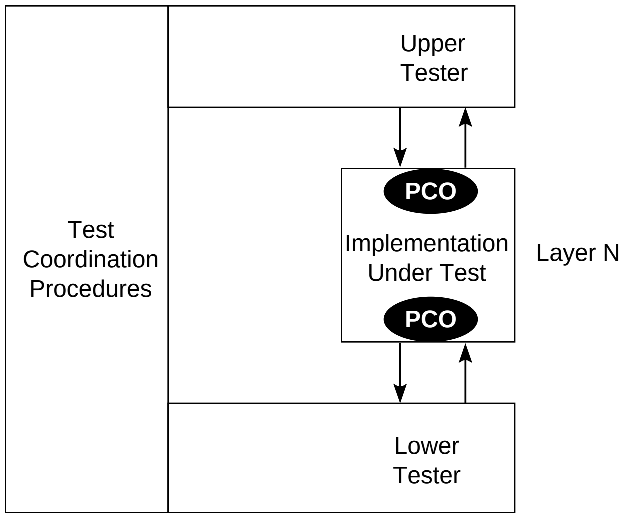


Figure 13. Local method of protocol testing. A single layer of a protocol is surrounded by an upper tester and a lower tester, both coordinated by test coordination procedures.

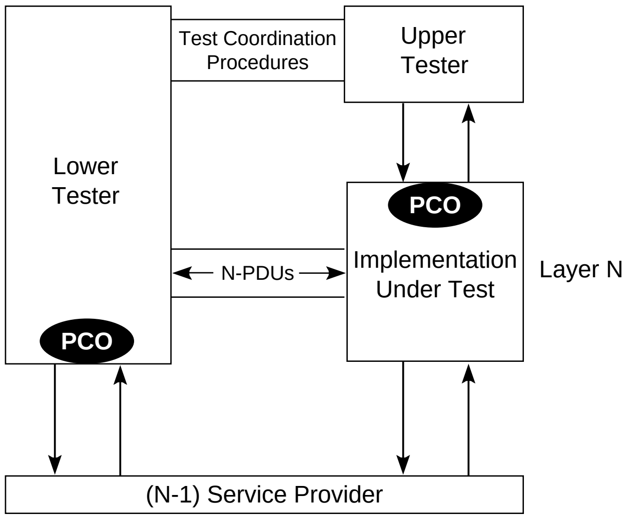


Figure 14. The distributed method. The lower tester runs on a different machine, and several other layers may intervene between the LT and the IUT.

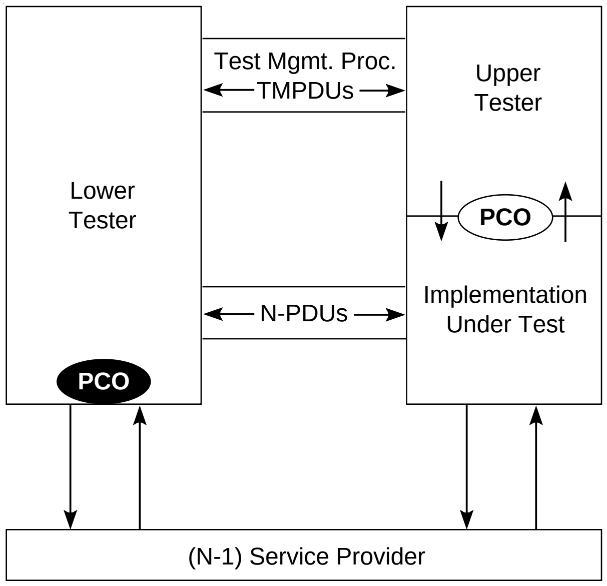


Figure 15. The coordinated method. Unlike the distributed method, this method does not assume an exposed interface between the UT and the IUT, and test coordination is automated.

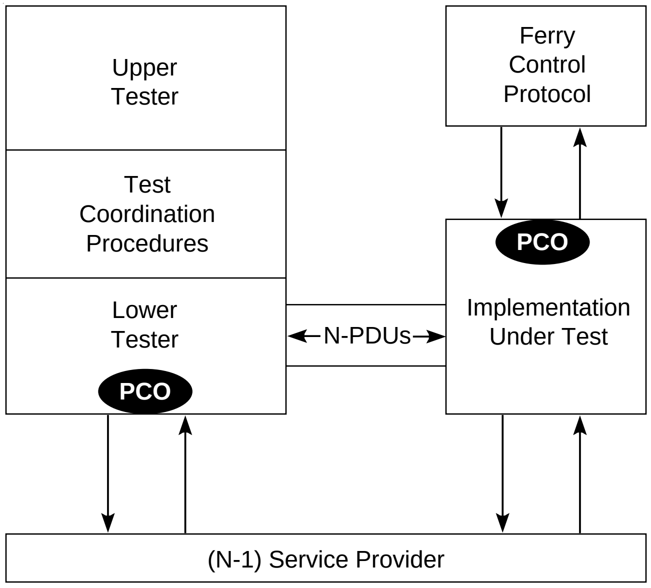


Figure 16. The ferry method. The upper and lower testers run on a remote machine, and a stub procedure called a ferry control protocol takes UT commands and exercises the upper PCO of the IUT.

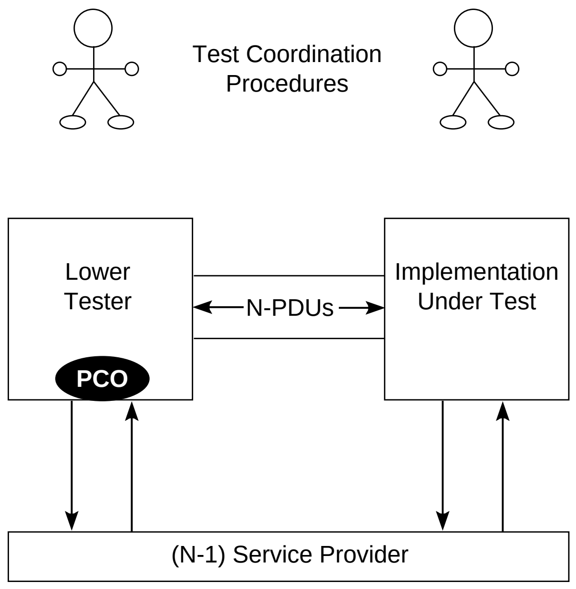


Figure 17. The remote method. No upper interface of the IUT is assumed, and no explicit test coordination procedures are assumed.

The distinguishing sequence method employs choices of input/output sequences from the IUT that unambiguously determine its internal state, where the IUT is modeled as an FSM. The W method regards the IUT as an FSM and selects two sets of input sequences. The P set traverses partial paths through the state diagram regarded as a tree, while the W set, also called the *characterization* set, differentiates between each pair of terminal or leaf states. Test sequences are built by concatenating sequences from P with sequences from W. In the transition tour method, the control flow diagram of the IUT modeled as an FSM is traversed to generate input sequences. The UIO method is a refinement of the DS method that views the IUT modeled as a directed graph of an FSM and finds the unique input sequences that produce unique output sequences. The RCP method is an extension of the UIO method produced by optimizing the choice and order of test sequences for minimum cost (i.e., execution time). Each of the methods mentioned have strengths and drawbacks that are beyond the scope of this report.

Interesting recent work (Tripathy and Sarikaya 1991, Katsuyama et al 1991, Naik and Sarikaya 1992) has concentrated on generating test cases from the formal description used to validate the protocol specification. This approach has the potential for increased coverage and increased accuracy due to greater mechanization and the use of the same FDT used to specify the protocol originally.

4.2.2 Appropriate Use of Communication Services

Even if a communications protocol is well designed, the application software using its services may do so poorly. This may vary from unnecessary use of communication services (inefficiency) to failure to deal with error conditions reported by protocol software (the equivalent of an unspecified reception).

Inefficiency potentially affects reliability because inefficient systems perform nonessential acts, increasing hazard exposure by increasing the number of functions executed. Protocols by necessity deal with errors in a general way. Application software (e.g., the reactor protection software) is the level at which appropriate responses to communication failures should be chosen. The problem in using the services of a communication protocol in safety-critical applications is to select a minimal but complete set of services that exactly addresses the system communication requirements.

For highly reliable, relatively simple systems such as reactor protection systems (RPSs), it may be possible to extend the FSM description of lower-level protocols to the RPS software itself, in effect considering the RPS software as the top-level protocol of a protocol stack. This would accrue the benefit of including the RPS description in the FDT description with associated specification validation, and the inappropriate usage question would be moot. The parallels between RPS operation and communication protocols make this an attractive idea.

Alternatively, modeling and simulation techniques exist (Sagoo and Holding 1991, Peterson 1977) that can be used for combined modeling of the communications system and the RPS software usage of communication services. Certain safety properties of the combined model can be proved using reachability analysis in a manner analogous to that mentioned in Section 4.2.1.2, Validation of Protocol Specifications.

4.2.3 Effects of Performance on Reliability

Performance (i.e., speed) affects reliability because the communication system may become congested and actually fail to deliver some messages or because real-time deadlines may not be met. Performance problems are sometimes hard to separate from inappropriate-use problems since the latter often overload an otherwise acceptable protocol. However, when system architecture has been adjusted to minimize extra "hops,"[22] minimal communications services are used to accomplish system objectives, and the system is still too slow, there is a performance problem.

As was pointed out in Section 2.6.1, Factors That Limit Performance, diagnosis of performance problems can be complicated and is beyond the scope of this work. See Preckshot (Preckshot 1992) for assistance on complexity and scaling problems. At least one graphical software tool is available for network performance analysis (Van Zijl et al 1992), and more general analysis tools using Petri nets have been reported (Conte and Caselli 1991, Sagoo and Holding 1991). Engineering for performance as part of system requirements is covered in Lawrence, Smith, and Smith (Lawrence 1992b, Smith 1990, Smith 1992).

4.3 Techniques and Tools

Organized development of computer network protocols began in the late 1960s and has continued unabated. Techniques were *ad hoc*, as were the rest of software development methods, through the 1970s, with more formal approaches beginning in the late 1970s. Since then there has been a steady increase in techniques and tools available for protocol synthesis, validation, and verification, and for network design. A sampling of significant tools and techniques follows.

4.3.1 Formal Description Techniques (FDTs)

Formal Description Techniques (FDTs) are languages in which communication protocol specifications can be expressed unambiguously. They are a prerequisite for validation. Two international standards organizations, ISO and CCITT, have taken the lead in standardizing FDTs, followed by the national standards bodies of many participating countries (the U.S.A. among them).

[22] A "hop" is one message transmission over exactly one physical link. Multihop messages pass over several physical links to get from source to destination. This sometimes happens even though a single direct physical link exists between source and destination because the message must be passed through an intermediary program.

4.3.2 Petri Nets

Petri nets (Peterson 1977) have been used in protocol engineering for specification, validation, and simulation of communication protocols (King 1991). They are also used for real-time system specification (Sagoo and Holding 1991) and for modeling general distributed systems (Conte and Caselli 1991).

4.3.3 SDL

SDL (Specification and Description Language) (CCITT 1988, Katsuyama et al 1991) is a language that allows the description of protocols as extended finite-state machines and was developed under CCITT auspices. The language description exists in a CCITT recommendation (Z.100), and it is used in the FOREST test environment mentioned in Section 4.3.14.

4.3.4 LOTOS

LOTOS (Language of Temporal Ordering Specification) (Bolognesi and Brinksman 1987) is one of two FDTs developed by ISO. LOTOS is based on *process algebra*, and describes process behaviors and interactions in an extension of the ideas proposed by Milner (Milner 1983) and Hoare (Hoare 1978). This FDT is oriented toward temporal relations between externally observable behavior of systems or system parts.

4.3.5 Estelle

Estelle (Extended State Transition Language) (Budkowski and Dembinski 1987) is the second ISO-developed FDT and owes acknowledgment to SDL. Estelle is a Pascal-based[23] FDT oriented toward finite-state machines. The system to be modeled is structured as a hierarchy of possibly asynchronous tasks.

4.3.6 ASN.1

ASN.1 (Abstract Syntax Notation One) (ISO 1988a, ISO 1988b) is a language used for data structure specification. The separation of ASN.1 from other behaviorally or functionally oriented FDTs reflects a logical partition between how protocols work versus the format of the data they exchange to accomplish their work. ASN.1 is useful during verification of particular implementations of protocols, when precise data format becomes important.

4.3.7 Z

Z (pronounced "zed" by most practitioners) (Spivey 1989, Williams 1992) is a formal method language used for general formal method software development, and it has also been used for communication protocol development (King 1991). Z is a more generally oriented FDT that is not specifically directed toward finite-state machines or communicating sequential processes (Hoare 1978) (see LOTOS, above).

4.3.8 Others

Other FDTs or formal methods have been reported in use developing communication protocols. VDM (Vienna Development Method), CCS (Calculus of Communicating Systems), and Object-Z (Holzmann 1992, King 1991) are three such languages.

---

23 Pascal is a high-level computer programming language normally used for sequential computer programs.

4.3.9 TTCN

TTCN (Tree and Tabular Combined Notation) (Katsuyama 1991, ISO 1989, Rayner 1987) is a language for abstract test specification. TTCN was designed so that test specifications could be written with sufficient generality that different implementations of the same validated protocol specification could be tested from the same abstract testing description.

4.3.10 Test Generation from LOTOS

A method for test generation from LOTOS specifications has been described (Tripathy and Sarikaya 1991) that was implemented on Sun workstations. There is no report of commercial availability of this tool.

4.3.11 Escort

Escort (Wakahara et al 1989) is a UNIX implementation of a protocol synthesis and validation tool using the methods described by Kakuda and Saito (Kakuda and Saito 1991). The tool uses the SDL FDT for protocol specification and provides an interactive synthesis-validation environment.

4.3.12 KDSS/KSPS

Artificial Intelligence techniques (AI) have been applied to protocol design (Shiratori et al 1992). A collection of rule-based inference-engine expert systems (KDSS) is under development with a protocol synthesis part (KSPS) running on Sun 3 workstations. This system uses an N = 2 finite-state machine model and expands the global state interactively as the user synthesizes the protocol (see Section 4.2.1, *Protocol Software*).

4.3.13 GMA

GMA is a graphical-user-interface modeling tool for analyzing performance of communication networks (Van Zijl et al 1992). This tool takes graphical descriptions of networks, routing, link, and node information and uses various queuing theory models to compute expected network performance. GMA is modular so that other theoretical performance models can be added to the model-solution kernel, thereby supporting different queue service strategies (i.e., different policies for handling messages at each node).

4.3.14 FOREST

FOREST (FORmal Environment for Systematic Testing) (Katsuyama et al 1991) is a combination tool composed of four subparts that together implement the ISO model for layered conformance testing (Rayner 1987, Linn 1989). FOREST uses the ISO FDTs SDL and ASN.1 and the ISO abstract test language TTCN to generate test cases, execute tests, and report results. The ISO FDT Estelle will be supported in the future.

References

Abramson, Norman, “The Aloha System—Another Alternative for Computer Communications,” Fall Joint Computer Conference, 1970, pp. 281–285.

Baran, P., “On Distributed Communications,” Rand Corp. Memo RM-3420-PR, August 1964.

Beeby, William D., “The Heart of Integration: a Sound Data Base,” *IEEE Spectrum*, May 1983, pp. 44–48.

Bertine, H. V., “Physical Level Protocols,” *IEEE Trans. on Communications*, COM-28, No. 4, April 1980, pp. 433–444.

Bochman, G. V., and Sunshine, C., “Use of Formal Methods in Communication Protocol Design,” *IEEE Transactions on Communications*, COM-28, 4(April 1980), pp. 624–631.

Boeing Corporation, *Technical and Office Protocols*, Version 1.0, November 1985.

Bolognesi, Tommaso, and Brinksman, Ed, “Introduction to the ISO Specification Language LOTOS,” *Computer Networks and ISDN Systems*, 14(1987), pp. 25–59.

Brand, Daniel, and Zafiropulo, Pitro, “On Communicating Finite-State Machines,” *J. ACM*, Vol. 30, No. 2, April 1983, pp. 323–342.

Budkowski, S., and Dembinski, P., “An Introduction to Estelle: A Specification Language for Distributed Systems,” *Computer Networks and ISDN Systems*, 14 (1987), pp. 3–23.

Burr, W. E., “The FDDI Optical Data Link,” *IEEE Communications Magazine*, Vol. 24, No. 5, May 1986, pp. 18–23.

Carlson, David E., “Bit-Oriented Data Link Control Procedures,” *IEEE Trans. on Communications*, COM-28, No. 4, April 1980, pp. 455–467.

Carrera, John P., and Easter, James R., “Advanced Alarm Management in the AWARE System,” *IEEE Nuclear Science Symposium and Medical Imaging Conference*, Sante Fe, NM, Nov. 2–9, 1991, pp. 1389–1393.

“Functional Specification and Description Language,” *Recommendation Z.100*, CCITT, 1988.

Ciminiera, L., Demartini, C., and Valenzano, A., “Performance Analysis of Type 3 LLC in Industrial 802.5 Networks,” *IEEE Network*, Vol. 2, No. 3, May 1988, pp. 90–96.

Clapp, George H., “LAN Interconnection Across SMDS,” *IEEE Network Magazine*, September 1991, pp. 25–32.

Conard, James W., “Character-Oriented Data Link Control Protocols,” *IEEE Trans. on Communications*, COM-28, No. 4, April 1980, pp. 445–454.

Conte, Gianni, and Caselli, Stefano, “Generalized Stochastic Petri Nets and Their Use in Modeling Distributed Architectures,” *IEEE Conference CH301-5/91/0000/0296*, 1991, pp. 296–303.

Davies, D. W., Barber, D. L. A., Price, W. L., and Solomonides, C. M., *Computer Networks and Their Protocols*, John Wiley & Sons, New York, 1979.

Defense Technical Information Center, *DOD Standard Internet Protocol*, January 1980a, RFC 760, IEN 128.

Defense Technical Information Center, *DOD Standard Transmission Control Protocol*, January 1980b, RFC 761, IEN 129.

Defense Technical Information Center, *User Datagram Protocol,* August 1980c, RFC 768, IEN 129.

Folts, Harold C., “X.25 Transaction-Oriented Features—Datagram and Fast Select,” *IEEE Transactions on Communications*, COM-28, No. 4, April 1980, pp. 496–500.

General Motors Corporation, *General Motors Manufacturing Automation Protocol, Version 2.1*, March 31, 1985.

Habara, Kohei, ISDN: "A Look at the Future Through the Past," *IEEE Communications Magazine*, November 1988, pp. 25–32.

Hoare, C. A. R., "Communicating Sequential Processes," *Comm. ACM 21*, 8 (August 1978) 666–677.

Holzmann, Gerard J., "Protocol Design: Redefining the State of the Art," *IEEE Software*, January 1992, pp. 17–22.

The Institute of Electrical and Electronic Engineers, *IEEE Standard Glossary of Software Engineering Terminology*, IEEE Std. 610.12-1990, corrected edition of February, 1991, ISBN 1-55937-067-X.

Inose, Hiroshi, and Saito, Tadao, "Theoretical Aspects in the Analysis and Synthesis of Packet Communication Networks," *Proceedings of the IEEE*, Vol. 66, No. 11, November 1978, pp. 1409–1422.

International Organization for Standardization (ISO), "Reference Model of Open Systems Interconnections," *Document no. ISO/TC97/SC16 N227*, June 1979.

"Information Processing Systems—Open Systems Interconnection—Specification of Abstract Syntax Notation One (ASN.1)," *ISO 8824*, 1988a.

"Information Processing Systems—Open Systems Interconnection—Specification of Encoding Rules for Abstract Syntax Notation One (ASN.1)," *ISO 8825*, 1988b.

"Information Processing Systems—OSI Conformance Testing Methodology and Framework," *ISO/DIS 9646*, 1989.

Kakuda, Yoshiaki, and Saito, Hironori, "An Integrated Approach to Design of Protocol Specifications Using Protocol Validation and Synthesis," *IEEE Transactions on Computers*, Vol. 40, No. 4, April 1991, pp. 459–467.

Katsuyama, Kotaro, Sato, Fumiaki, Nakakawaji, Tetsuo, and Mazuno, Tadanori, "Strategic Testing Environment with Formal Description Techniques," *IEEE Transactions on Computers*, Vol. 40, No. 4, April 1991, pp. 514–525.

King, Paul W., "Formalization of Protocol Engineering Concepts," *IEEE Transactions on Computers*, Vol.40, No. 4, April 1991, pp. 387–403.

Kleinrock, Leonard, "Principles and Lessons in Packet Communications," *Proceedings of the IEEE*, Vol. 66, No. 11, November 1978, pp. 1320–1329.

Kleinrock, Leonard, and Tobagi, Fouad A., "Packet Switching in Radio Channels: Part 1—Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics," *IEEE Trans. on Communications*, Com-23, No. 12, December 1975, pp. 1400–1416.

Lawrence, Dennis J., Subject: *Software reliability guidance*, Lawrence Livermore National Laboratory, to be released. 1992a.

Lawrence, Dennis J., Subject: *Software complexity and performance guidance*, Lawrence Livermore National Laboratory, to be released. 1992b.

Lin, Fuchun Joseph, and Liu, Ming T., "Protocol Validation for Large-Scale Applications," *IEEE Software*, January 1992, pp. 23–26.

Linn, Richard J., Jr., "Conformance Evaluation Methodology and Protocol Testing," *IEEE Journal on Selected Areas in Communications*, Vol. 7, No. 7, September 1989, pp. 1143–1158.

Malek, Manu, "Integrated Voice and Data Communications Overview," *IEEE Communications Magazine*, Vol. 26, No. 6, June 1988, pp. 5–15.

Manber, Udi, "Chain Reactions in Networks," *Computer*, October 1990, pp. 57–63.

McGuffin, L. J., Reid, L. O., Sparks, S. R., "MAP/TOP in CIM Distributed Computing," *IEEE Network*, Vol. 1, No. 3, May 1988, pp. 23–31.

McNamara, John E., *Technical Aspects of Data Communication*, Digital Press, Bedford MA, 1977.

Metcalfe, Robert M., and Boggs, David R., "Ethernet: Distributed Packet Switching for Local Computer Networks," *Comm. ACM 19*, 7 July 1976, pp. 395–404.

Milner, R., "Calculi for Synchrony and Asynchrony," *Theoretical Computer Science 25*, 1983, pp. 267–310.

Modiri, Nasser, "The ISO Reference Model Entities," *IEEE Network Magazine*, July 1991, pp. 24–33.

Mollenauer, James F., "Standards for Metropolitan Area Networks," *IEEE Communications Magazine*, Vol. 26, No. 4, April 1988, pp. 15–19.

Mortazavi, Behzad, "Performance of MAP in the Remote Operation of a CNC," *IEEE Trans. on Software Engineering*, Vol. 16, No. 2, February 1990, pp. 231–237.

Naik, Kshirasagar, and Behcet, Sarikaya, "Testing Communication Protocols," *IEEE Software*, January 1992, pp. 27–37.

Parnas, David L., "Designing Software for Ease of Extension and Contraction," *IEEE Trans. on Software Engineering*, Vol. SE-5, No. 2, March 1979, pp. 128–138.

Peterson, James L., Petri Nets, *ACM Computing Surveys*, Vol. 9, No. 3, September 1977, pp. 223–252.

Postel, Jonathon B., "An Informal Comparison of Three Protocols," *Computer Networks 3 (1979)*, pp. 67–76.

Postel, Jonathon B., Sunshine, Carl A., Cohen, Danny, "The ARPA Internet Protocol," *Computer Networks 5 (1981)*, pp. 261–271.

Pouzin, Louis, and Zimmermann, Hubert, "A Tutorial on Protocols," *Proceedings of the IEEE*, Vol. 66, No. 11, November 1978, pp. 1346–1370.

Preckshot, George G., *Real-Time Systems Complexity and Scalability*, Lawrence Livermore National Laboratory, to be released.

Probert, Robert L., and Saleh, Kassem, "Synthesis of Communication Protocols: Survey and Assessment," *IEEE Transactions on Computers*, Vol. 40, No. 4, April 1991, pp. 468–476.

Rayner, D., "OSI Conformance Testing," *Computer Networks and ISDN Systems*, 14(1987), pp. 79–98.

Roberts, Lawrence G., "The Evolution of Packet Switching," *Proceedings of the IEEE*, Vol. 66, No. 11, November 1978, pp. 1307–1313.

Ross, Floyd E., "FDDI—A Tutorial," *IEEE Communications Magazine*, Vol. 24, No. 5, May 1986, pp. 10–17.

Ross, Floyd E., "An Overview of FDDI: The Fiber Distributed Data Interface," *IEEE Journal on Selected Areas in Communications*, Vol. 7, No. 7, September 1989, pp. 1043–1051.

Rybczynski, Antony, "X.25 Interface and End-to-End Virtual Circuit Service Characteristics," *IEEE Transactions on Communications*, COM-28, No. 4, April 1980, pp. 500–510.

Sagoo, J. S., and Holding, D. J., "The Use of Temporal Petri Nets in the Specification and Design of Systems with Safety Implications," *IFAC Algorithms and Architectures for Real-Time Control*, Bangor, North Wales, UK, 1991.

Saleh, B. E. A., and Teich, M. C., *Fundamentals of Photonics*, John Wiley & Sons, Inc., New York, 1991.

Shiratori, Norio, Takahashi, Kaoru, Sugawara, Kenji, and Kinoshita, Tetsuo, "Using Artificial Intelligence in Communication System Design," *IEEE Software*, January 1992, pp. 38–46.

Shoch, John F., and Hupp, Jon A., "Measured Performance of an Ethernet Local Network," *Comm. ACM 23*, 12, December 1980, pp. 711–721.

Smith, Connie U., *Performance Engineering of Software Systems*, Addison-Wesley Publishing Company, 1990.

Smith, Connie U., Subject: *Performance engineering of safety-critical systems*, Lawrence Livermore National Laboratory, to be released. 1992.

Sparkman, Debra R., *Techniques, Processes, and Measures for Software Safety and Reliability*, UCRL 108725, Lawrence Livermore National Laboratory, April 30, 1992.

Spivey, J. M., *The Z Notation: A Reference Manual*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

Sproull, Robert F., and Cohen, Dan, "High-Level Protocols," *Proceedings of the IEEE*, Vol. 66, No. 11, November 1978, pp. 1371–1386.

Strayer, W. Timothy, and Weaver, Alfred C., "Performance Measurements of Data Transfer Services in MAP," *IEEE Network*, Vol. 2, No. 3, May 1988, pp. 75–81.

Stuck, Bart W., "Calculating the Maximum Mean Data Rate in Local Area Networks," *Computer*, May 1983, pp. 72–76.

Tanenbaum, Andrew S., “Network Protocols,” *Computing Surveys 13*, 4 December 1981, pp. 453–489.

Tobagi, Fouad A., “Multiaccess Protocols in Packet Communication Systems,” *IEEE Trans. on Communications*, Vol. COM-28, No. 4, April 1980, pp. 468–488.

Tobagi, Fouad A., Gerla, Mario, Peebles, Richard W., Manning, Eric G., “Modeling and Measurement Techniques in Packet Communication Networks,” *Proceedings of the IEEE*, Vol. 66, No. 11, November 1978, pp. 1423–1447.

Tripathy, Piyu, and Sarikaya, Beheet, “Test Generation from LOTOS Specifications,” *IEEE Transactions on Computers*, Vol. 40, No. 4, April 1991, pp. 543–552.

Van Zijl, Lynette, Mitton, Deon, and Crosby, Simon, “A Tool for Graphical Network Modeling and Analysis,” *IEEE Software*, January 1992, pp. 47–54.

Wakahara, Yasushi, Kakuda, Yoshiaki, Ito, Atsushi, and Utsunomiya, Eiji, “Escort: An Environment for Specifying Communication Requirements,” *IEEE Software*, March 1989, pp. 38–43.

Watson, George F., “Faults & Failures,” *IEEE Spectrum*, May 1992, p. 52.

Williams, Lloyd G., *Formal Methods in the Development of Safety Critical Software Systems*, UCRL 109416, Lawrence Livermore National Laboratory, April 3, 1992.

Wirbel, Loring, “Conflicting Efforts Are Plaguing FDDI,” *Electronic Engineering Times*, December 1991.

Xerox Corp., Digital Equipment Corp., and Intel Corp., *The Ethernet, A Local Area Network, Data Link Layer and Physical Layer Specifications*, Version 1.0, September 30, 1980.

Zimmermann, Hubert, “OSI Reference Model—The ISO Model of Architecture for Open Systems Interconnection,” *IEEE Trans. on Communications*, Vol. COM-28, No. 4, April 1980, pp. 425–432.

Zucconi, L., and Barter, R., *Verification and Validation Techniques for Software Safety and Reliability*, Lawrence Livermore National Laboratory, to be released.

Glossary

ADCCP. Advanced Data Communication Control Procedures (ANSI X3.66-1979) is an HDLC-like link-level serial communication standard.

ANSI. American National Standards Institute. An organization that develops industry standards. Best known for its development of the ASCII code set and, later, the ANSI code set. The latter is a superset of the former.

ARPANET. Advanced Research Projects Agency Network.

Bandwidth. The information-carrying capacity of a communications channel (the difference, expressed in Hertz, between the highest and lowest frequencies of a band).

Bit. A binary digit (0 or 1). The smallest unit of data communications information.

Bit-stuffing. Frame-oriented protocols transmit variable-length sequences of bits demarcated by uncounterfeitable frame marker bit subsequences (frames). Frame markers typically are a sequence of six one-bits (e.g., 01111110), and no valid data sequence is allowed to have more than five consecutive one-bits. This is accomplished by having the hardware serializer (or deserializer) "stuff" (or "unstuff") a zero-bit after any five one-bits in transmitted (or received) data. The frame is therefore started and ended by frame markers and bytes are synchronized at the beginning of each frame. In transmission and reception hardware performs marker generation (or stripping) and bit stuffing (or unstuffing), and the frame is delivered to software protocol layers synchronized and unsullied by non-transparent link-control characters.

Block. (1) A set of things, such as words, characters, or digits, handled as a unit. (2) A group of contiguous characters formed for transmission purposes. Blocks are separated by interblock gaps and each block may contain one or more records. The groups are separated by interblock characters. Definition (2) is most often applied to data stored on magnetic tape.

Byte. A set of contiguous bits, sometimes adhering to a certain standard code (ASCII, ANSI, or EBCDIC) to represent a certain alphabetic, numeric, or symbolic character. Usually 8 bits in length. Also known as an octet when it has 8 bits.

CCITT. International Telegraph and Telephone Consultative Committee. An international standards group for data and voice communications, composed primarily of the national telephone companies of various countries.

Channel. (1) A path along which signals can be sent; for example, data channel, output channel. (2) A band of frequencies. (3) The portion of a storage medium that is accessible to a given reading station.

Character. (1) An elementary mark or event that may be combined with others, usually in the form of a linear string, to form data or represent information. (2) One of a set of elementary symbols which normally include both alpha and numeric codes plus punctuation marks and any other symbol which may be read, stored or written.

CIM. Computer Integrated Manufacturing. Jargon for one or more computer systems that provide design tools, accounting tools, management tools, manufacturing planning tools, and manufacturing automation, and which are connected together so that various manufacturing and business activities can share information easily.

Coaxial cable. An electrical cable consisting of a center conductor surrounded by a cylindrical dielectric that is in turn surrounded by a cylindrical outer conductor. The outer conductor may be braid or foil and may be enclosed in an insulating sheath. Found in some local area networks (such as Ethernet) and in IBM's local 3270 connections. It has become a popular medium in networks because of its large capacity, low error rates, and configuration flexibility. Materials such as Teflon and PVC are used as dielectrics. There are different types; thin Ethernet uses RG58, and 3270 uses RG62.

Common carrier. In telecommunications, an agency that, for a fee, provides communication services by any of a number of modes—wire, radio, optical, or other electromagnetic systems. Examples include the local telephone company, MCI, GTE, and AT&T.

CMISE. Common Management Information Service Environment, an OSI model Level-7 protocol set for network and system management.

CSMA. Carrier Sense Multiple Access, an access method for controlling contention on a communications medium that has multiple transmitters connected to it. Transmitters wishing to transmit listen for a clear channel before transmitting.

CSMA/CD. Carrier Sense Multiple Access with Collision Detection, an access method for controlling contention on a communications medium that has multiple transmitters connected to it. Transmitters wishing to transmit listen for a clear channel before transmitting, and continue listening during transmission to detect collisions. If a collision occurs, the colliding transmitters abort transmission and try again later.

Datagram. A short message analogous to a telegram. Datagrams are exchanged in "connectionless" service, where no prior association has been made between sender and receiver.

Deadlock. Deadlock is, simply described, the situation where protocol entities await each other because each has something the other wants.

Deterministic. A "deterministic" communication system delivers messages within a finite, predictable time delay that is a function of system communication load. Determinism is considered to be a very important quality for communication systems used in real-time applications. Unfortunately, no communication system is deterministic under all conditions, and unpredictable delays will be incurred for at least some errors or failures.

Differential driver. Electronic circuits that apply electrical signals to a two-wire communication medium so that the voltage on one wire moves in opposition to the voltage on the other. Contrast with single-ended driver.

DQDB. Distributed Queued Dual Bus, the access method and architectural arrangement used in the IEEE 802.6 Metropolitan Area Network standard.

EIA. Electronics Industries Association. U.S. manufacturers' organization responsible for the "RS-" or Recommended Standards such as RS-232-C.

Encapsulation. Encapsulation is the idea that protocols at succeeding lower levels of the OSI model wrap or encapsulate data coming down from higher levels with information that the lower protocols need to perform their functions.

EPA. Enhanced Performance Architecture, also called miniMAP. A MAP subset consisting only of OSI model layers 1, 2, and 7. This subset was devised because of performance concerns about the full MAP used in local automation cells.

Ethernet. One of the first local area network schemes commercially available, Ethernet was unveiled in the late 1970s by Digital Equipment Corporation, Intel, and Xerox as a method for connecting minicomputers and office products. The IEEE 802.3 standard is essentially Ethernet: a 10-Mbit/second baseband coaxial-cable-based local area network that uses CSMA/CD as the access control method. Thick Ethernet is the original standard: it uses a fat coaxial cable, often yellow. Thin Ethernet is a later standard that achieves the same performance over shorter distances; it uses a smaller, more flexible, less expensive coaxial cable. Implementations of Ethernet over optical fiber and twisted pair cable are also available.

Event. (1) Change of state of one or more inputs. (2) An interrupt in a computer system.

FDDI. Fiber Distributed Data Interface, a 100-Mbit/second network standard, originally developed for optical fiber.

Fiber-optic cable. Consists of optical fibers made of plastic or glass. Very high performance: bandwidth up to 3.3 billion Hertz vs. 500 million Hertz for coaxial; data rates of over one billion bits per second; very low error rates; unaffected by electrical or electromagnetic interference; very small and light. In fiber-optic communications, electrical signals are translated into light pulses by a modulator, transmitted over the fiber by a light source, and detected and converted back into electrical signals by photoelectric diodes. There are fiber-optic cards that work in place of Ethernet cards.

Frame. (1) A set of consecutive digit time slots in which the position of each digit time slot can be identified by reference to a framing signal. (2) A variable-length sequence of bits demarcated by uncounterfeitable frame marker bit subsequences.

Frame marker. Frame markers typically are a sequence of six one-bits (e.g., 01111110), and no valid data sequence is allowed to have more than five consecutive one-bits. Other frame markers are possible, such as link control symbols in N of M bit coding schemes (used in the FDDI link-level standard).

FTAM. File Transfer, Access, and Management. An OSI model Level-7 protocol for file operations across networks.

HDLC. High-level Data Link Control. ISO protocol for data exchange.

Host. (1) The main computer in a data processing system, or the main processing unit in a multi-unit system (often a mini or mainframe). (2) Any computer attached to a computer communication network.

IEEE. Institute of Electrical and Electronics Engineers. Among the primary duties of the institute is to develop industry standards. The IEEE Local-Network Group (the Project 802 Committee) decided to develop several different local-network standards at the Data Link and Physical Layers (see ISO). Key standards include:

Data Link Layer

802.2—The Data Link Layer protocols used in conjunction with the Physical-Layer standards. 802.2 is the Data Link layer to be used with 802.3,802.4, 802.5, and 802.9, which are physical-layer standards.

Physical-Layer Standards

802.3—Virtually identical to the Ethernet standard (allows various speeds). Includes thin Ethernet, StarLAN, and broadband Ethernet.

802.4—A broadband token bus standard intended for use in factories.

802.5—Virtually identical to the IBM Token Ring.

802.6—Metropolitan area network

802.9—A voice/data standard similar to Northern Telecom's LanStar, for transmission over twisted pair cable.

ISDN. Integrated Services Digital Network. A series of international standards for all-digital telephone systems, in which voice and data are mixed at the telephone set.

ISO. International Organization for Standardization. In 1978 ISO issued a recommendation for greater conformity in the design of communications networks and the control of distributed processing. They developed a model—the Open System Interconnection (OSI) model—that describes a hierarchical structure for data communication through networks. In this seven-layer model for network architecture, each layer provides a certain subset of services to the overall network functions. The layers are handled by a combination of hardware and software. From highest to lowest level each layer is described below.

Application/User Layer—Services are provided that directly support user and application tasks and overall system management (i.e., resource sharing, file transfers, remote file access, database management, and network management). The application layer provides entry points (subroutine calls) to access these services.

Presentation Layer—Negotiates data interchange and format conversion so that incompatible data types in different hosts can be reconciled.

Session Layer—Establishes and controls system-dependent aspects of communications sessions between specific nodes in the network. This bridges the gap between the services provided by the transport layer and the logical functions running under the operating system in a participating node.

Transport Layer—Provides end-to-end transport of data once a path has been established, allowing nodes to exchange data reliably and sequentially, independent of which systems are communicating or the routing through intervening networks in the path.

Network Control Layer—Addresses messages, sets up the path between communicating nodes, routes messages across intervening nodes to their destination, and controls the flow of messages between nodes.

Data Link Layer—Establishes an error-free communications path between network nodes over a physical channel, frames messages for transmission in packets, checks integrity of received messages, manages access to and use of the channel, and may ensure proper sequence of transmitted data if connection-oriented service is selected.

Physical Link Layer—Defines the electrical and mechanical aspects of interfacing to a physical medium for transmitting data, as well as setting up, maintaining, and disconnecting physical links.

LAN. Local Area Network.

LAP. Link Access Procedure, a CCITT HDLC-like link-level communication standard.

LAPB. Balanced Link Access Procedure, a CCITT specification for point-to-point communication using HDLC frames and underlying the CCITT X.25 virtual circuit recommendation (what the CCITT calls its standards).

Layering. A strategy for modularizing and separating computer network protocols into a series of layers, best illustrated by the ISO OSI seven-layer model. Layering has two advantages: separating functions and providing well-defined interfaces between layers.

Link. A channel or circuit designed to be connected in tandem with other channels or circuits. In automatic switching, a link is a path between two units of switching apparatus within a central office.

Liveness. The quality a protocol must provide. This generally means that the protocol cannot "hang up" because it must provide a minimal set of functions no matter what, hence the sobriquet "liveness."

LLC. Logical Link Control. The upper sublayer of two sublayers making up Level 2 (the link layer) in the OSI model. The LLC sublayer was first standardized by the IEEE 802 committee and contains the media-independent part of the link level protocol.

MAC. Media Access Control. The lower sublayer of OSI Data Link, Level 2, as defined by IEEE 802 standards. The FDDI standard is a MAC sublayer standard.

MAN. Metropolitan Area Network, IEEE standard 802.6. This is a MAC sublayer.

MAP/TOP. Manufacturing Automation Protocol/Technical Office Protocol. A GM/Boeing-sponsored pair of standards for factory/office automation.

Master–slave. A link-level access control procedure in which one node, the master, orchestrates access by all other nodes. Typically, this is done by polling the slave nodes in round-robin fashion.

MiniMAP. Also called Enhanced Performance Architecture (EPA). A MAP subset consisting only of OSI model layers 1, 2, and 7. This subset was devised because of performance concerns about the full MAP used in local automation cells.

MMS. Manufacturing Message Service. An OSI-model Level-7 protocol to allow manufacturing cell supervisory computers to manage various computer-controlled tools.

Modularity. The characteristic of being self-contained, with limited and well-defined interfaces to other parts of a system.

Modulation. The adjustment of carrier waves according to the information to be transferred. The three most common forms of modulation are amplitude, frequency, and phase.

Multibus. An Intel Corporation backplane computer bus designed for its 80x86 computers and later standardized because of its ubiquity.

Multiple-access. In computer communications, the ability of a network to allow more than two nodes to communicate over a single medium, although not necessarily simultaneously. The procedure for ensuring orderly access (and thus preventing chaos) is called an "access control procedure," and such procedures are the subjects of many communication standards.

Multiplexing. (1) To interleave or simultaneously transmit two or more messages on a single channel. (2) The combining of two or more signals into a single wave (the multiplex wave) from which the signals can be individually recovered. (3) An inherent feature in many computer communication protocols that permits many apparently simultaneous data streams to flow between computer network nodes over a relatively small number of physical links.

Node. (1) A terminal of any branch of a network or a terminal common to two or more branches of a network. (2) A computer connected to a computer network.

Noise. Unwanted disturbances superposed upon a useful signal that tend to obscure its information content.

Octet. An 8-bit byte. Used by the datacom industry and cited in many standards.

Packet. Discrete, uninterrupted units of data into which long messages are divided for transmission over a network. They can be of fixed or variable length, but they generally have a specified maximum length (e.g., 128 bytes or 1,500 bytes). Packets usually contain bits for synchronization, control information, message number, destination and source addresses, acknowledgment, error checking, and the data itself.

PDU. Protocol Data Unit. The data, including headers and other protocol information, that is transferred as a unit from one layer to another in the OSI model or across a network link.

Physical-level protocol. Defines the electrical and mechanical aspects of interfacing to a physical medium for transmitting data, as well as setting up, maintaining, and disconnecting physical links.

Protocol. A set of rules that must be observed to ensure an orderly and comprehensible information exchange between different nodes on a network. The three most common in the Ethernet world are XNS (Xerox Network System), TCP/IP (Transmission Control Procedure/Internet Protocol), and DECnet. 3COM uses XNS; Sun workstations use TCP/IP; and VAX computers running VMS use DECnet.

Protocol stack. A series of protocols conforming to a layered model, such as the ISO OSI seven-layer model.

Public data network. Data transmission services supplied in the form of a network either by a commercial interest or a government agency.

SDLC. Synchronous Data Link Control, an IBM-originated serial communication standard that is almost identical to HDLC.

Single-ended driver. Electronic circuits that apply electrical signals to one wire of a two-wire communication medium. The second wire is grounded or common. Contrast with differential driver.

State. The values of a minimal set of functions, of which values contain information about the history of a system sufficient to determine the future behavior, given knowledge of future inputs.

Switched circuit. Essentially a telephone circuit. A circuit made up by switching wires connected between two terminals, i.e., telephones. The salient feature is that the entire bandwidth of the connection is dedicated to the terminals during the time of the connection. See also "virtual circuit."

Synchronization. The term "synchronization" is overloaded in electrical engineering and computer science, which may reflect its importance. To place the term in perspective, its meanings in a variety of contexts are given:

Communications—A means of ensuring that both transmitting and receiving stations are operating together (in phase). Maintaining phase lock between a multiple of the frequency of a carrier or subcarrier and demodulation circuitry in a receiver.

Computer science/operating systems—Queuing an event or data item so that it can be handled later by software that is already busy doing something. When the handler software is ready to service a new request, it asks for the next event or data.

Computer science/distributed systems—Maintaining time or event order at physically separated points in a distributed system.

TCP/IP. Transmission Control Protocol/Internet Protocol. A standard suite of network protocols created by the Department of Defense. These protocols are in wide use in technical environments. Often comes bundled with computers running the UNIX operating system. It is a robust suite that deals with internetworking among various vendor's equipment.

Teletype. A teleprinter (mechanical typewriter actuated by electrical signals coming from a serial communications channel) made by the Teletype Corporation.

TOP. Technical Office Protocol. Developed by Boeing Corporation to support their database-oriented computer-aided design and engineering activities for designing jetliners.

Transparency. A capability of a communications medium to pass within specified limits a range of signals having one or more defined properties; for example, a channel may be code transparent, or an equipment may be bit-pattern transparent.

Twisted pair. A cable composed of two small insulated conductors, twisted together without a common covering. The two conductors of a twisted pair are usually substantially insulated, so that the combination is a special case of a cord. Telephones are wired with unshielded twisted pair cable.

Virtual circuit. The analog of a physically switched circuit, but implemented in a packet communications system. The virtual circuit provides the illusion of a physically switched circuit at the end points, but no physical channel is dedicated entirely to the two respondents. Instead, conversations between various respondents are packet-multiplexed over available communication links. The best-known virtual circuit protocol is the CCITT X.25 protocol.

VME bus. A Motorola backplane bus standard originally designed for the 68000 and now IEEE 1014. The VMEbus specification includes physical dimensions (often called Eurocard) for cards that plug into the backplane. VMEbus is a 32-bit, asynchronous, non-multiplexed bus. Several bus masters are possible, and a 32-bit address range is possible, with 8-, 16-, or 32-bit data transfers. The bus has two connectors, P1 and P2, with 96-pin DIN connectors, and supplies +5V and ±12V.